Published by The Internet-First University Press

The Early Years of Academic Computing: A Collection of Memoirs

Edited by William Y. Arms and Kenneth M. King

This is a collection of reflections/memoirs concerning the early years of academic computing, emphasizing the period from the 1950s to the 1990s when universities developed their own computing environments.

Cornell University
Initial Release: June 2014
Increment 01: April 2016
Increment 02-04: April 2017

NOTE: This is the first part of a set of memoirs to be published by The Internet-First University Press.

This book – http://hdl.handle.net/1813/36810
Books and Articles Collection – http://ecommons.library.cornell.edu/handle/1813/63
The Internet-First University Press – http://ecommons.library.cornell.edu/handle/1813/62

An Incremental Book

Publisher's Note

This is the second Internet-First publication in a new publishing genre, an '**Incremental Book**' that becomes feasible due to the Internet. Unlike paper-first book publishing – in which each contribution is typically held in abeyance until all parts are complete, and only then presented to the reading public as part of a single entity – with Internet publishing, book segments (*Increment*s) can be released when finalized. Book parts may be published as they become available. We anticipate releasing updated editions from time-to-time (using dated, rather than numbered editions) that incorporate these increments. These digital collections may be freely downloaded for personal use, or may be ordered as bound copies, at user expense, from Cornell Business Services (CBS) Digital Services by sending an e-mail to digital@cornell.edu or calling 607.255.2524. In the body of the message include the URL for this book or increment, and request contact regarding payment.

As these increments become available online, the eCommons@Cornell digital repository will automatically provide a listing of the most recent additions to this collection. Furthermore, we will provide a chronologically ordered PDF file named '**Increments and Forthcoming Sections**' to be added to this eCommons collection. From time-to-time these 'increments' will be integrated into the book, which will then bear a new edition date.

The Internet-First University Press
June 2014

copy editor and proofreader: Dianne Ferriss
Published by The Internet-First University Press
Ithaca, New York
Co-founders: J. Robert Cooke and Kenneth M. King

Preface

In March 2014 Bill Arms approached Ken King about publishing his memoir about the *Early Years of Academic Computing* in the Internet First University Press (IFUP). The IFUP had been created in 2003 by Bob Cooke and Ken King as part of a project to encourage open publication on the Internet as a first choice for scholarly publication. Bill had written his memoir as a result of his observation that todays students were unaware of the many contributions Universities had made to creating the current computing environment and some of those contributions needed to be documented. Ken King stated that he believed that Universities deserved far more credit for computing advances than they had received.

Ken suggested that Bill try to create an "incremental book" by persuading other pioneers to contribute their memoirs. The incremental book idea was developed by Bob Cooke as part of the IFUP effort to get scholarly information on the Internet. The idea was that the IFUP would publish chapters or sections in a book on the Internet as they are finished, rather than wait until the whole book had been completed. Bill suggested to Ken that he develop his own memoir and that he and Ken then jointly approach other pioneers as co-editors to create an incremental book composed of memoirs by people who labored at Universities and may recall events and contributions that helped change the world. Ken agreed and the second section in our book covers his observations about life in the trenches at Universities during the period between 1953 and 1993. The book that we hope to create is possibly an effort not unlike the story about an elephant touched by blind men. People that we hope will contribute will certainly have different slants on major events at the Universities that helped change the world.

Contents of this Incremental Book

01

ARMS

The Early Years of Academic Computing:

A Memoir by William Y. Arms

This is from a collection of reflections/memoirs concerning the early years of academic computing, emphasizing the period from the 1950s to the 1990s when universities developed their own computing environments.

Initial Release: June 2014

Published by The Internet-First University Press
Copy editor and proofreader: Dianne Ferriss
The entire incremental book is openly available at http://hdl.handle.net/1813/36810

Books and Articles Collection – http://ecommons.library.cornell.edu/handle/1813/63
The Internet-First University Press – http://ecommons.library.cornell.edu/handle/1813/62

A. Memoir by William Y. Arms

Preface

The past fifty years have seen university computing move from a fringe activity to a central part of academic life. Today's college seniors never knew a world without personal computers, networks, and the web. Sometimes I ask Cornell students, "When my wife and I were undergraduates at Oxford, there were separate men's and women's colleges. If we wanted to meet in the evening, how did we communicate?" They never knew a world without telephones and email, let alone smartphones, Google, and Facebook.

They are also unaware that much of modern computing was developed by universities. Universities were the pioneers in end-user computing. The idea that everybody is a computer user is quite recent. Many organizations still do not allow their staff to choose their own computers, decide how to use them, and select the applications to run. But end-user computing has been established in universities since the 1960s. When the computer industry was slow in seeing the potential, universities took the initiative. This narrative describes a thirty-year period when academic computing diverged from the mainstream. Universities built their own state-of-the-art systems and ran them successfully. They led the development of timesharing and local area (campus) networks. They were major contributors to distributed computing, email, the national networks, and the web.

This is not a history. It is a memoir. As a student, faculty member, and administrator, I lived through many of these developments. From 1978 to 1995 I was in charge of computing at two of the leading universities, Dartmouth College and Carnegie Mellon. After 1995 I was no longer personally involved in the developments, but in the final section I describe how academic computing and the mainstream have now come together again, to the benefit of both.

There is a glaring gap that runs throughout this narrative. It says little about the impact of computing on academic life, on teaching, research, and libraries. The final section has a few thoughts about educational computing, but it is a huge topic that deserves a much fuller treatment.

Disruptive change is an underlying theme of this period. I have tried to give some indication of what it was like to be a computing director, the opportunities and the excitement, but also the pressures and uncertainties. I was fortunate in working for two remarkable presidents, John Kemeny at Dartmouth and Richard Cyert at Carnegie Mellon. Each was a visionary in seeing how computing could be used in higher education. When I remember my colleagues at other universities wrestling with less enlightened leaders, I know how lucky I was.

Contents

Chapter 1

The Early Days

A second generation computer

This photograph is of a Ferranti Pegasus computer in the Science Museum in London. The museum claims that it is the oldest computer in the world that is still operational.

Wikipedia

Background: The 1960s

The 1960s

The mid-1960s were a pivotal time for computing. In 1965, as semiconductors replaced vacuum tubes, Gordon Moore of Intel observed that the number of transistors on integrated circuits was doubling every two years. A year earlier, IBM had announced the 360 family of computers. Timesharing was born that same year, when two people at Dartmouth College using separate terminals typed "PRINT 2+2" and (after some bugs were fixed) both got the answer 4.

Moore's Law is not a law of nature, but for fifty years hardware engineers have used it as a target. On aggregate they have actually done better than Moore predicted. This exponential growth has created enormous opportunities, but it has also created intense challenges. Many years later, Allen Newell pointed out that collectively we do well in predicting the performance of components such as processors and disks; we do quite well in predicting the next generation of hardware, such as networks and personal computers; but we have consistently failed to predict the breakthrough applications enabled by this hardware. He might have added that we also fail to anticipate the organizational and social impact.

A Pegasus computer

My early experience provides some typical examples of computing in the 1960s, before the emergence of timesharing. It was characteristic of the period that as a mathematics undergraduate at Oxford, graduating in 1966, I never went near a computer.

I wrote one small program in 1962 while at school in England. A local engineering company had a Ferranti Pegasus computer and invited a group of us (all boys) to visit it. There is a Pegasus in the Science Museum in London. It is an imposing collection of shining boxes, which occupy a space the size of a small room, but the computer we saw had most of its covers removed exposing racks of vacuum tubes and electrical circuits. The covers were off because the components were highly unreliable. The processor was an array of modules, each about the size of a shoe box and containing several vacuum tubes. Individual modules could be slid out of the rack and replaced, even while the machine was powered up.

We each had one attempt to run a small program. The programming language was Pegasus Autocode, a dialect of the family that later became Algol. My program solved a simple mathematical puzzle. We wrote our program instructions on paper. A clerk keyed them on to 5-track paper tape and an operator fed the tape into the computer. After a substantial pause, the results were printed out on a teletype terminal. My program gave the correct result. Other people were not so successful.

Fortran programming with punched cards

After my brief experience at school, I did not use a computer again until 1966 when I was a master's student at the London School of Economics. The school had an IBM 1440 computer.

The programming language was Fortran II, which was easy to use for the mathematical programs that we wrote. Nobody gave us instructions. Programming was assumed to be a simple skill that any student could pick up from the manual.

Before class we left our coding sheets at the reception desk and an hour later we picked up the results. We got back a deck of cards, a listing of our program, its compilation, and results. If we wanted to make changes, we wrote them on a new coding sheet and handed back the deck of cards. Apparently the school had decided that it was less expensive to pay somebody to key our programs than to waste expensive computer time trying to run badly keyed programs that we had punched ourselves.

English Electric - Leo - Marconi

My first job after graduation was in the operational research group of English Electric - Leo - Marconi. As the name implied, this was a merger of three separate companies, each of which manufactured computers during the 1960s. By the time that I actually started work the company had changed its name to English Electric - Leo, and soon afterwards it merged with International Computers and Tabulators Ltd. to become International Computers Limited (ICL).

A Leo computer from the early 1960s

Leo was one of the pioneers of computer-based data processing. It began as an in-house venture of J. Lyons, a bakery and food company. Lyons built the original machines for its own use. Leo was perhaps the first company to use multiprogramming successfully. The Leo III was able to run up to three programs simultaneously. The company also developed a high-level programming language, Cleo, similar to IBM's Cobol.

LEO Computers Society

In 1967, English Electric - Leo introduced a new range of computers, known as System 4, which was a derivative of the RCA Spectra series. These machines used the same instruction set as the recently introduced IBM 360 series. The company was proud of the semiconductor technology, claiming to be the first commercial computer to have more than one gate on a silicon chip. Building on experience with the Leo 3, the central processor had hardware support for multiprogramming.

For the first three months on my new job, I was sent on an intensive training course to learn about these new machines. The course included assembly level programming and went into considerable detail about the hardware. This was a fortunate time to learn about computing as I was trained in the computer architecture that was to dominate mainframe computers for many years.

The IBM 360 and its Clones

Third generation computers

The computers introduced in the mid-1960s, such as the IBM 360 family, were called "third generation" computers. Third generation computers were important because each computer in a family had the same architecture. Previously, every new range of computers had had a different architecture and machines within a range were often incompatible. Whenever customers bought a new machine they had to convert all their old programs. The huge suites of software that we run today would not be possible without stable architectures and standardized programming languages. Over the fifty years since its first release, the IBM 360 architecture has been steadily enhanced, such as when virtual memory was added, but the enhancements have generally been backwards compatible.

An IBM 360

An IBM System/360 in use at Volkswagen.

German Federal Archives

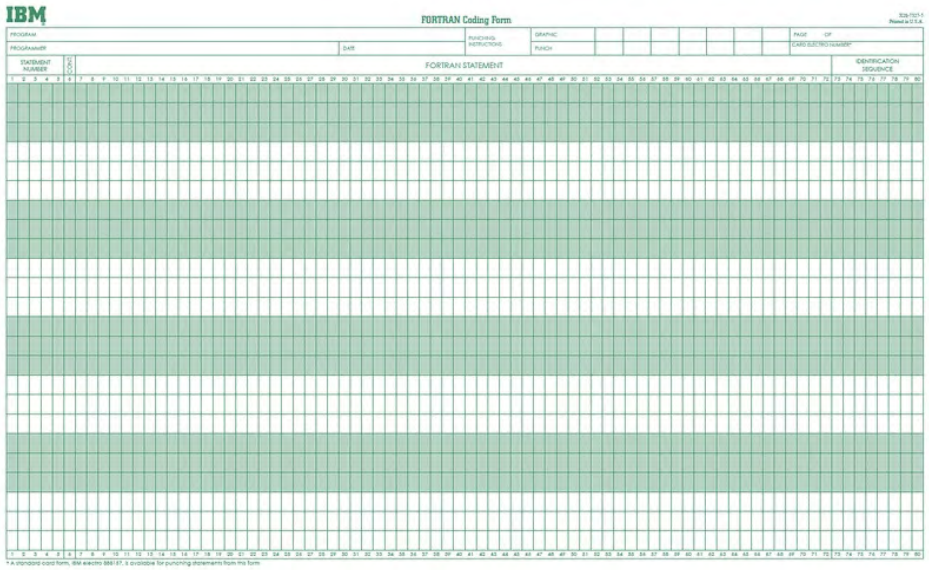
A third generation computer consisted of a group of large metal boxes. The cabinets were laid out in an air-conditioned machine room and connected with heavy-duty cables under a raised floor. Photographs show machine rooms that were models of tidiness, but in practice a machine room was a busy place, full of boxes of punched cards, magnetic tapes, and stacks of paper for the line printers. They were noisy, with card readers and line printers chattering away, and the continuous rush of air-conditioning.

Solid-state components had replaced vacuum tubes, but the logic boards were still large and the central processor was several tall racks of circuit boards with interconnecting cables. Memory used magnetic cores, which were bulky and expensive. In 1968, the price of memory was about $1 per byte.

Most of the boxes in the machine room were peripherals and their controllers. Rotating disks were beginning to come on the market, but most backup storage was on open-reel magnetic tape. The tape held nine bits of data in a row across the tape, representing one byte plus a digital check. Early drives wrote 800 rows per inch, but higher densities followed steadily. To sort data that is on magnetic tape requires at least three tape units and most systems had many more. The controllers that connected the tape and disk units to the central processor were imposing pieces of expensive hardware.

The architecture of the IBM 360 supported both scientific computing and data processing. It firmly established the eight-bit byte as the unit of storage for a character and four bytes were organized as a word. Earlier computers had used a variety of word sizes and characters had often been stored as six bits. In the days before virtual memory, memory was organized into 4096-byte modules and I vividly remember the challenges of memory management when writing assembly code programs.

The instruction set was designed to support high-level languages, such as Cobol for data processing, and Fortran for scientific work. Arithmetic could be decimal, operating on character strings, or binary, operating on words that could represent either integer or floating-point numbers. By using variable-length instruction formats, complex operations could be carried out in a single instruction. For example, only one instruction was needed to move a string of characters to another location in memory. Apart from the memory management, I found the assembly code straightforward and easy to program in.



A Fortran coding sheet

Programmers wrote their code on coding sheets. A clerk keyed the program onto punched cards, one card for each line of the program.

Wikipedia

In the days before terminals and interactive computing, a programmer wrote the instructions on coding sheets, with one 80-character line for each instruction. The instructions were punched onto cards, one card per instruction. A colored job control card was placed at the head of the card deck and another at the end, followed by data cards. The complete deck was placed in a tray and handed to the computer operators. Sometime later the programmer would receive a printout with the results of the compilation and the output of any test runs. Computer time was always scarce and one test run per day was considered good service. This placed a premium on checking programs thoroughly before submission, as even a simple syntax error could waste a day's run.

Commercial data processing on third generation computers such as the IBM 360 used batch processing to automate tasks such as billing, payroll, stock control, etc. The algorithms resembled the modern "map/reduce" methods that are used to build indexes for Internet search engines. These methods are used to process huge volumes of data, such as when building indexes for Internet search engines. These algorithms are used when the amount of data is so large relative to the memory of the computer that there is no possibility of holding comprehensive data structures in memory and random access processing would be impossibly slow. Therefore the processing methods use sequential processing. The basic processing steps are sorting and merging, and the computation is done on small groups of records that have the same key.

In earlier years, some data input had used paper tape, but by the late 1960s punched cards were standard. An advantage of punched cards was that an error could be corrected by replacing a card, though it was easy to make further mistakes in doing so. Every computer center had a data preparation staff of young women who punched the transactions onto punched cards in fixed format. The cards were verified by having a second person key the same data. Data preparation was such a boring job that staff turnover was a perpetual problem. The British driving license system, which we developed at English Electric - Leo, was always understaffed in spring before the school year came to an end and the next wave of school leavers could be hired.

The first step in processing a batch of cards was to feed them into a card reader. A "data vet" program checked the cards for syntax errors and wrote the card images onto magnetic tape. This was then sorted so that the records were in the same sequence as the master file.

Records were stored in a master file on magnetic tape and sorted by an identifying number, such as a customer number. Sorting records on magnetic tape by repeated merging was time consuming. The performance depended on the number of tape drives and the amount of memory available. Time was saved by having tape drives that could be read backwards, so that no time was wasted rewinding them.

The master file update program would typically be run at the end of the day. The previous version of the file would be merged with the incoming transactions and an updated version of the master file written on a new magnetic tape. The update programs generated copious output to be printed, including business transactions such as bills or checks, management reports, and data errors. This output was written to another tape, which was then sorted by the category of printing, so that the operators could load the various types of paper into the printers as required.

The standard printer was a line printer, usually uppercase only. The paper was fan-folded with sprocket holes on the side and was advanced one line at a time. Most manufacturers used drum printers, with a complete set of characters around the drum, but IBM used a chain printer in which the characters were on a moving chain. I never understood how it worked. With either type of printer the actual printing was caused by a flying hammer that pressed the paper against the metal character.

Every part of the operation was prone to failure. Most computer centers halted work for several hours every night for maintenance, but even so hardware and software failures were common. An advantage of batch processing was that the system could write a checkpoint at the end of each batch. If the card reader jammed or a magnetic tape failed, the operators went back to the previous checkpoint, and reloaded the cards or the previous copy of the tape and restarted the job.

University Computing before Timesharing

End-user computing

During the 1960s, academic computing diverged from the mainstream. Most commercial applications, whether data processing or scientific, were large production jobs that ran for several hours and used the entire computer. Companies hired professionals to write the programs, punch the input, and run the jobs. In universities, the faculty, students, and researchers wrote their own programs and often ran them themselves. They would spend long periods developing programs, hoping for fast turnaround of compilations and small tests, followed by a few large computations. Since many of the programs were run only a few times, priority was given to convenient program development.

Computer hardware was improving rapidly, both in performance and reliability, but hardware was a scarce resource. Processor cycles could not be wasted. Because there were very great economies of scale, the best strategy for a university was to buy a single large computer and find a way to share it among the users. This led to the growth of university computer centers.

The University of Sussex

When we were at the University of Sussex from 1969 to 1972, we had computing facilities that were typical of the better universities at the time. The university computer was an ICL 1904A. The ICL 1900 series was an early third generation system that competed quite successfully with the IBM 360. The architecture used a 24-bit word. I think that the computer at the University of Sussex had 32K words, equivalent to less than 100K bytes.

ICL developed a sequence of operating systems called George. While waiting for ICL to deliver George III, the University of Sussex developed a simple monitor for running small Fortran jobs. It used a circular buffer on magnetic disk that held jobs waiting to be processed. The jobs were card decks of Fortran programs to be compiled and run. Since card reading was slow, the aim was always to have several jobs that had been read into the buffer, waiting to be run, so that the central processor was never idle. Output to the line printer was also buffered.

The Fortran compiler and the monitor used almost all the memory, but about 4K words were spare. My wife and I reached an agreement with the computing center that we could use this small amount of memory for our experiments with online catalogs. The computer had provision for a few terminals and we used one of them. This sounds an absurdly small amount of memory but we were able to use the Fortran IV overlay mechanism. This was a primitive form of paging by which the programmer specified that certain subroutines and data structures could overlay each other in memory. Since the basic Fortran I/O package itself used more than 4K, we wrote a physical I/O routine to control the terminal and never loaded the I/O package.

The university ran the computer for two shifts per day. The third shift, from midnight to 8 a.m., was available for researchers. Several of us went through the operator training course and for one night shift per week we had sole use of the machine. The procedure for rebooting the computer was typical of the era. Nowadays computers hold their boot program in some form of persistent storage, but the boot program for the 1904A was on paper tape. The first step in booting the machine was to set a few instructions using hand switches on the central processor. These instructions were executed when the computer was powered up. They instructed the paper tape reader to read the boot program into memory, and the boot program then read the operating system from magnetic tape.

Because the machine room was noisy, we would work on our programs in the reception area. We could tell what the machine was doing by listening to the audio monitor. This device, which was common on machines of that era, made a distinctive tone for each category of instruction that was being executed. Since each program

had a distinctive pattern of sounds we could tell when a job, such as a tape sort or the Fortran compiler, came to an end.

Punched card equipment

The central computer was used almost entirely for academic work. Administrative data processing used punched card equipment. For example, the library's circulation system used nothing but punched cards and made no use of the computer. For one of my analyses I had more than seventy trays each containing 2,000 cards.

The punched card machines were direct descendents of the Hollerith machines that were built in the early 1900s to tabulate census data. IBM took over the Hollerith company and much of the company's data processing expertise came from its experience with punched card equipment.

Punched card equipment

In this IBM publicity photograph the operators are men, but in my experience most of them were women. The tidiness is also misleading. In practice, trays of cards and boxes of printer paper were stacked everywhere.

IBM Archives

The data processing room at the University of Sussex had about six large devices, each with its specialized function: a card sorter, copier, collator, tabulator, printer, etc. The collator was particularly important as it could merge data from two stacks of cards and punch out a new card, which combined the information from them. Each device was controlled by cables that were inserted into a plug board, thus creating a very simple program.

As an example, the card sorter had one input hopper and ten output hoppers. Sorting was one column at a time. The operator would use the plug board to specify a column of the card and other parameters. She would load the cards into the input hopper, one tray at a time, and the sorter would send each card to the output hopper that corresponded to the number punched in the appropriate column. To sort by a three-digit number, the cards would be passed through the sorter three times, sorting first by the least significant digit. Complex data processing operations, such as a master file update, were carried out by passing trays of cards repeatedly through the various devices.

Chapter 2

Timesharing

Two pioneers of educational computing

Tom Kurtz (left) and John Kemeny were pioneers in using interactive computing for undergraduate education. They developed the Basic programming language and the Dartmouth Time Sharing System.

Dartmouth College Library

Early Timesharing

Multiprograming, multitasking, and timesharing

Mainframe computers were expensive. The central computer that Dartmouth bought in 1975 cost more than four million dollars, a great deal of money for a small university. Moreover, these computers had huge economies of scale. Several years later, when Cornell studied the options for replacing its mainframe computer, an IBM 370/65 cost twice as much as a smaller IBM 370/55, yet was four times as powerful. With such economies of scale, every organization followed the same strategy: buy the largest computer you could afford and find ways to maximize the amount of work done on it.

Computer hardware improved rapidly during the 1960s, both in performance and reliability, but processor cycles were a scarce resource not to be wasted. Computers of the previous generation had run one program at a time and the central processor might spend much of its time idling. For example, a program to process a tray of punched cards would spend most of its time waiting for the next card to be read.

By the middle of the decade, computers were powerful enough to run more than one program at a time. The first technique that allowed several tasks to be run simultaneously was called multiprogramming. Each program was given a priority. At any given moment the operating system would offer the central processor to the highest priority program. Suppose that the highest priority program copied punched cards onto tape. The program would start to read a card and then wait while the card was read. This caused an interrupt and the scheduler would start up the next program in priority. This might send data to a line printer. This program too would spend much of its time waiting and the central processor could be assigned to another program. Rather unintuitively, the highest priority was given to the program that used the central processor least. This primitive form of multiprogramming required the operators to understand the characteristics of each program. It evolved into multitasking, where the user specifies the characteristics of a job, such as memory requirements, and the operating system schedules the execution. Modern operating systems use preemptive multitasking, where the operating system manages the entire scheduling, interrupting (or preempting) tasks after a predetermined time interval.

IBM's OS/360, released in 1966, is often considered to be the first modern operating system. Its focus was on efficient batch processing for commercial data processing. The painful lessons learned during its development were the subject of the first book on software engineering, "*The Mythical Man Month*" by Fred Brooks.

The beginnings of timesharing

Batch processing operating systems, with their emphasis on long-running data processing jobs, were unsuitable for academic computing. Universities, therefore, developed timesharing to give their faculty and students direct access to a central computer. The early systems included Multics at MIT, Titan at Cambridge University, and the Dartmouth Time Sharing System (DTSS). MIT and Cambridge emphasized the needs of researchers, while Dartmouth's focus was on undergraduate education. My first glimpse at timesharing was in the summer of 1965 when I visited Dartmouth and sat in on a lecture by John Kemeny on Basic programming. A year later, when my wife was a graduate student at MIT, Multics was in production and she was able to write Fortran programs for a research project.

Titan

This picture is of the operators' area in the Titan machine room at Cambridge University. Notice the hard-copy terminal with paper coming out.

University of Cambridge

Timesharing allows a central computer to be shared by a large number of users sitting at terminals. Each program in turn is given use of the central processor for a fixed period of time. When the time is up, the program

is interrupted and the next program resumes execution. This is called "time slicing." A program can also be interrupted before the end of the time slice, for example it might have to wait for input. Timesharing is particularly effective when each user wants to use the computer intermittently, such as for program development, which has long periods of editing followed by short test runs. It is not suitable for the batch processing that is typical of commercial data processing

Dartmouth's DTSS and MIT's Multics had different objectives, and the systems reflected these differences, but the architectures had many features in common. Multics was closely linked to the emerging discipline of computer science. It is often described as the precursor to Unix. I recall a visit from Brian Kernigham of Bell Labs to Dartmouth in about 1982 when he was intrigued to discover that features of Unix, such as pipes, had equivalents in both architectures. Most of Multics was written in a high-level language, PL/1, unlike other operating systems that were written in assembly code and tightly linked to the hardware architecture.

DTSS had a direct commercial impact. Since large computers were so expensive, commercial timesharing bureaus sold computer time to remote customers. The market leader was General Electric's GEIS, which was originally developed in a joint project at Dartmouth. Concepts from DTSS were adopted by the early minicomputer systems from companies such as Hewlett-Packard and Digital Equipment Corporation. Even the computer HAL in the film "2001: A Space Odyssey" had commands taken from DTSS.

Timesharing dominated academic computing until the late 1980s, when it was replaced by personal computers. Even then, the differences between academic and commercial computing remained and led universities to build the first large networks of personal computers. As a result academic computing and the computing mainstream followed separate paths for about thirty years, from the mid-1960s to the 1990s.

Basic programming

Basic was developed at Dartmouth as a straightforward programming language for students and faculty. It became the language of choice in educational computing. Basic was never suitable for large programs, and for this reason it is often dismissed by computer scientists, but it was ideally suited for its purpose. The simple syntax meant that it was easy for beginners. For example, no distinction was made between integers and floating point; they were just numbers. As editing was always a problem on hardcopy terminals, each Basic statement had a line number; to make a change, the user retyped the line. Finally, the syntax was carefully designed for very fast compilation. Dartmouth specialized in fast compilers but almost every other version of Basic was interpreted.

Because of its efficient use of hardware, the first commercial timesharing systems were based on Basic. Later, for the same reasons, Basic was the dominant language on the early personal computers. Microsoft's first product was a version of Basic. Unfortunately for Basic's reputation, Microsoft Basic had numerous crude extensions to give the programmer control of the hardware.

As computers became more powerful, Dartmouth steadily extended the language. The final version was an elegant structured language, with an effective choice of procedures, and good vector graphics. The design choices always emphasized simplicity. In the early 1980s we used Basic for the introductory computer science course at Dartmouth and PL/1 for the second. For the equivalent courses at Cornell today, we use Python and Java.

Commercial timesharing systems

Some of the first minicomputers ran small timesharing systems with a dedicated Basic interpreter. In 1972 I moved to the British Open University, which was the pioneer in distance education providing degree programs for students across Britain. The university had a national timeshared network using Hewlett-Packard's HP 2000 Time-Shared Basic. The university had teletype terminals in study centers around the country. They

were connected to the computer system by placing an ordinary telephone handset into an acoustic coupler. The transmission speed was 110 bits per second. The Hewlett-Packard computers could each support 32 simultaneous users. There was one computer at the Open University's headquarters in Milton Keynes, and I think that there was a second system at a center in the north of England. The computer provided only one service, a Basic editor and interpreter. The version of Basic and the command language were essentially the same as Dartmouth's first version.

The first two computing courses that we introduced at the Open University were based on this system, which was a great success. The main difficulty was not technical. Many of the university's students lived long distances from the study centers and we had to design the courses so that the students did not have to visit the centers very often. My wife was one of the first people to have a terminal at home, in the kitchen. In those days, it was the ultimate status symbol of the working wife.

Digital Equipment Corporation (often known as Digital or DEC) created three very different timesharing systems, each of which was widely used in universities: RSTS for the PDP 11, TOPS-20 for the DEC-20, and VAX/VMS. The original RSTS was another Basic system, rather like Hewlett-Packard's, but the final version, for the PDP 11/70, was a general-purpose timesharing system. It supported up to 63 users and was used by smaller colleges into the 1990s. I never used RSTS, but both the Open University and Carnegie Mellon had DEC-20s. Carnegie Mellon had six of them in the central machine room and the computer science department had several more. TOPS-20 had a flexible command language, with a good balance between simplicity and generality. Unfortunately the unreliability of the hardware often let down the excellence of the software. The DEC-20s had an extended version of Basic but Digital's particular strength was its Fortran 77 language. The VAX built its reputation on the fast computation for science and engineering, but the VMS operating system was also good for timesharing. Many liberal arts colleges used VAX/VMS for their academic computing, and in the 1980s it was widely used for departmental computing centers.

Finally, Unix began as a timesharing system at Bell Labs, where it ran on a variety of Digital minicomputers. The version that became a central part of academic computing was the Berkeley Software Distribution (BSD), from the University of California at Berkeley. The definitive version was BSD 4.2 for Digital VAX computers, released in 1983.

Timesharing at Dartmouth

The Dartmouth Time Sharing System

The founders of computing at Dartmouth were two mathematicians, Tom Kurtz and John Kemeny. With help from a team of undergraduates, they created the Basic programming language and the Dartmouth Time Sharing System (DTSS). Kemeny later became President of Dartmouth and Kurtz was the first director of the Kiewit computing center. Under his leadership, Kiewit gained a reputation for outstanding service to the academic community.

I was on sabbatical at Dartmouth in 1976 to 77, and from 1978 to 1985 I was head of computing, with various job titles. When I came to Dartmouth, the Kiewit technical group was very strong. It is hard to single out individuals, but Stan Dunten (networking) and Phil Koch (operating systems and programming languages) were outstanding. I was fortunate to inherit a smooth-running computer center. The key people were Tom Byrne, who ran the business side, and Punch Taylor, who had overall responsibility for the technical work.

DTSS reached its maturity during these years and it is interesting to compare the services that it provided with the networks of personal computers that swept universities soon afterwards. From a modern viewpoint the

most surprising feature was that the entire software was developed at Dartmouth. From the device drivers to the applications programs everything was written locally, either at Dartmouth or by DTSS, Inc., the commercial spin off. Most of the original system was written by undergraduates, and much of the maintenance was still done by student system programmers.

A teletype terminal

Early timesharing used hardcopy terminals of the type that were used for sending telegrams. They are often called "teletype" terminals, the name of one popular model, but they actually came from several vendors. They were replaced by the much superior DECwriters in the mid-1970s. Terminals cost between $1,000 and $1,500.

Wikipedia

DTSS began at a time when almost all academic computing used programs written by faculty and students for their own work. For this reason, the operating environment was tailored to compile and run small programs very quickly. Dartmouth provided excellent compilers for PL/1 and for Basic, which reached maturity during the late 1970s. To support up to 200 simultaneous users, the timesharing monitor enforced strict limits on each user's CPU usage, memory, and disk storage, and the system used these resources very efficiently. For example, in the days before virtual memory, the runtime environment solved the problem of running large programs by automatically swapping procedures within a user's memory allocation.

In 1978, DTSS was running on a Honeywell 66/40 computer, the successor to the GE 625 and 635 computers on which it was developed. The hardware was unusual for a third generation mainframe computer in that it had two processors, each consisting of the central processing unit, the memory, and a multiplexor. Each of these six units had a large cabinet, filled with racks of logic boards. Hardware engineers were in constant attendance.

The performance of each processor was about 1 million instructions per second, and the total memory was 2 megabytes. There were about ten disk drives, each the size of a small washing machine. The total disk capacity eventually reached about one gigabyte, which served the entire university. The disk drives were unreliable and backup copies of the data were stored on magnetic tape. A full backup was made once a week with a daily incremental backup.

The communications architecture was an important reason that DTSS was able to support so many users. All terminal traffic was handled by two Honeywell 716 minicomputers that acted as terminal concentrators. To minimize the number of interruptions, they collected keystrokes and sent them to the central computers in batches, usually one line at a time.

By 1978, the standard terminal was a DECwriter. This was a hard copy device that ran at 300 bits per second over an ordinary telephone line. For graphics, we used the Tektronix 4010 family of terminals. These terminals painted a sequence of vectors on the screen that could be deleted only by wiping the entire screen blank. During my sabbatical year I wrote a graphical extension to the Basic system to display vector graphics. A variant of the syntax was incorporated into later versions of the Basic compiler.

Editing is a problem on a hard-copy terminal. The Basic programming language originally solved this problem by giving a line number to each statement and encouraging users to simply retype a line if it needed to be changed. More advanced users were provided with context editors, which made substitutions based on pattern matching.

Video terminals such as Digital's VT 52 and VT 100 came into widespread use about 1980, usually running at 1,200 bits per second. They were dumb terminals with no internal processing.

When they were used for full screen editing, each keystroke had to be processed before writing on the screen. If this processing was on the central computer, as was done by most commercial companies, the transmission and processing times could lead to a frustrating delay after each keystroke. At Dartmouth, we solved the problem of screen editing by building our own terminal, the Avatar, by adding a Z80 microprocessor and cache memory to a standard video terminal. In combination with a special editor on the central computer, the Reactor, this provided an excellent screen editor. In typical Dartmouth style, the hardware and software were both developed locally with Jim Perry as the leader. After I left Dartmouth, the Avatar software was ported to Macintosh and IBM personal computers, and the Redactor was ported to Digital's VAX/VMS.

As I write this description thirty-five years later, the system sounds primitive in the extreme, but many people considered that Dartmouth provided the best academic computing of any university in the world. The university was justifiably proud of its achievements. The technical staff was outstanding and Dartmouth was the benchmark for supporting users who had no interest in technicalities.

Computers in education

DTSS and Basic were simple and easy to use. Basic was never intended as a language for large programs, but was excellent for writing programs of a few hundred lines. Long before I arrived, the mathematics department passed a resolution that every student who took any mathematics course must learn to write simple programs. The department explicitly excused the faculty from this requirement, but many faculty members enjoyed writing programs to support their teaching and research.

These programs were made available through public libraries. There were libraries for specific courses and for disciplines such as statistics. I used to teach a course in computer graphics and had a course library with demonstration programs on topics such as splines and perspective. Other libraries included utilities such as text formatting and Dartmouth had a fine collection of games. Kemeny wrote a popular baseball simulation

based on the 1955 World Series triumph of the Brooklyn Dodgers. In anticipation of the modern "open source" movement, these programs were always stored as source code and continually upgraded by colleagues. Whenever we taught a course, it was a matter of pride to extend and improve the programs in the course library.

Dartmouth was unusual in that computing was seen primarily as a service to education and made no charges for the use of the computer. Funding followed what was called "the library model". Faculty and students were encouraged to use the computer. The center had an annual budget from the university and supplied a range of services at no cost to the user. I spent seven years successfully postponing demands from the central administration that we should introduce a charging scheme.

Kemeny and Kurtz's great achievement was to allow everybody to be a programmer. Modern computers provide a magnificent set of features, but modern user interfaces and networked communication are much more difficult to program. Many of today's faculty are skilled computer users, but it is much less usual to write a simple program the evening before a class.

The limitations of timesharing

Every computer system is a compromise. Flexibility and generality come at the price of simplicity. General purpose systems, such as IBM's MVS and Microsoft's Windows, are inevitably cumbersome. Timesharing's aim was to give users the illusion that they each had sole use of a powerful computer and for most users DTSS did this well. It was responsive and easy to use. Even under heavy load it allocated resources so efficiently that we never had to ration computing resources or charge users. But these virtues came at a price. Two important groups of academic users were poorly served.

Firstly, there was little support for the number crunching that is so important in science and engineering. DTSS's Fortran compiler was an afterthought, and there was no way to allocate large amounts of computer time to individual researchers, even if they had grant funds to pay for it. At a liberal arts university this was a serious problem and at a research university it would have been a fatal flaw. Many universities, where researchers and administrative computing shared an IBM mainframe, never adopted central timesharing.

Secondly, every program had to be written locally. Dartmouth built up an impressive public library, but academic computing was steadily moving away from program development to large commercial packages. The inability to run packages such as SPSS for statistics was one of the forces that led Dartmouth away from its own system.

For a while, Dartmouth addressed these two problems by augmenting the central timesharing system with minicomputers, which were used for number crunching and to run software packages, but minicomputers could not solve the fundamental weakness of timesharing, which was capacity. Timesharing was swamped by its own success. Every year more and more users wanted to run more complicated programs that required more terminals, more processor cycles, more memory, and more disk space. The central service could never keep up. Even the best systems slowed down when heavily loaded and made mockery of the illusion that the user had sole use of the computer.

When personal computers arrived there was no illusion and once powerful workstations became available timesharing was doomed. The capacity problems were solved by everybody buying their own computers. Timesharing systems, such as DTSS, became file servers and email hubs. Eventually they were replaced by server farms. But for twenty years timesharing was the leading edge of academic computing

Chapter 3

The Organization of Computing in Universities

A dinosaur from IBM

As personal computers replaced mainframes, the mainframes were colloquially referred to as "dinosaurs". This dinosaur was a gift from IBM after an advisory board meeting.

Photograph by William Arms

Departmental Computing Centers

Minicomputers

During the 1970s, the economies of scale in computer hardware were gradually reversed. I do not understand the engineering reasons that caused this, but it coincided with the development of large-scale integrated (LSI) semiconductors for processors and the phasing out of magnetic core memory. A new type of computer emerged, known as minicomputers, and a new group of computer companies. The market leader was Digital Equipment Corporation. By the late 1970s, minicomputers such as Digital's PDP 11/70, for timesharing, and VAX 11/780, for number crunching, were as cost-effective as the large central computers, and for some tasks were clearly superior

From the 1960s IBM had dominated mainframe computing in the United States, though other companies did well in certain markets, such as Burroughs in banking, and Univac and Control Data Corporation in scientific computing. The mainframe companies were popularly known as "IBM and the Seven Dwarfs" (Burroughs, Univac, NCR, Control Data Corporation, General Electric, RCA, and Honeywell). None of them foresaw the emergence of minicomputers and a new group of companies replaced them. For universities, Digital was the most important minicomputer company, but other companies such as Data General, Hewlett-Packard, Wang, and Prime were also successful. Data General's Nova was popular for computer graphics, and Wang was the leader in word processing and office automation. Many of these companies were spin-offs from MIT and scattered around Route 128 outside Boston.

For most of the 1970s the most widely used family of minicomputers was the Digital PDP 11. The smaller members of the family were used to control laboratory equipment and the larger ones could run a substantial time-shared system. At the end of the decade, 32-bit minicomputers such as Digital's VAX 11/780, the Data General Eclipse, and the Prime 750 had completely reversed the economies of scale. At Dartmouth, we had a Prime 750 for medium-scale number crunching and a variety of VAX and Prime computers for specific applications, such as the library's online catalog and the alumni database. At larger universities, research groups set up their own departmental computing centers. The first book to popularize the high-paced culture of computing was *The Soul of a New Machine* by Tracy Kidder, which described the rush to bring the Eclipse to market in 1980.

These minicomputers had excellent operating systems. While IBM's MVS operating system grew bigger and bigger until it became a nightmare to install and upgrade, the Digital and Prime systems were comparatively straightforward to manage. At Dartmouth we chose Prime because they supported the PL/1 programming language and the X.25 networking protocol, both of which we used for DTSS, but Digital's VAX with its VMS operating system proved more successful in the long term. Several members of the VMS group later moved to Microsoft where they were influential in developing the Windows operating systems.

Departmental computers

Minicomputers had a profound impact on the organization of university computing. For years, the management of central computers had assumed that hardware was a critical resource, particularly central processing cycles. To waste a single cycle was a crime. Only gradually did we come to realize that the time of the faculty and students was much more valuable. When I was on sabbatical at Dartmouth, I watched a group of mathematics professors who kept trying the same iteration, with ever-longer run times. Eventually they realized that they were iterating over a singular matrix and it would never converge. I was horrified at the waste of computer time until I realized that they had gained mathematical insights that were worth much more than the computer time. Not many years later I myself ran an integer programming problem for an entire weekend on a Prime 750. It would have been irresponsible to do so on a shared computer, but not on a machine that would have otherwise been idle.

When a university had a single central computer, the hardware determined the organization. A large central computer needed a team to manage it and to share its resources among departments and users. Almost every university created a computing center with its director. Most of these centers served both academic and administrative users. The center would have a system programming staff to support the central machine and applications programmers for administrative computing.

The centers never fully solved the problem that different groups of users have different computing needs. Most academic users run large numbers of small jobs, but some people want to run big computations or process very large sets of data. Administrative computing has an entirely different set of needs. In aggregate the capacity of the central computer was never enough to satisfy everybody. With varying success, computing centers attempted to balance priorities by technical, administrative, and financial mechanisms, but they could not make everybody happy.

For researchers with research grants, this unhappiness was aggravated by a peculiarity of how many universities charged for computer time. Research universities wanted to recover the cost of research computing from funding agencies, such as the National Science Foundation. These universities charged all users for machine time. Researchers used their grants to cover the costs, while other departments paid from their own budgets. To recoup as much money as possible from grants, the universities set their computing charges at the highest rate that the government would allow, including full overhead recovery.

Frustrated by these high charges and the inflexibility of central computing, the richer departments used their research grants to buy minicomputers and set up their own computing centers. Staff costs were largely eliminated by having graduate students look after the systems. When I arrived at Carnegie Mellon in 1985 the university claimed to have more VAXes than classrooms. There were well-run centers in computer science, electrical engineering, physics, statistics, and several other departments. Later, as personal computers became widely available, schools and colleges, such as fine arts and the business school, set up personal computing centers.

Many computer directors felt threatened by these developments. Some of these feelings were justified. The cost comparisons were often dubious, since departments did not include the indirect costs of running a computer center, which are substantial. The typical departmental computer was run by an assistant professor and a

graduate student. Only too often the student never graduated and the assistant professor did not get tenure, but the researchers clearly saw departmental centers as an effective way to spend their resources. An engineering professor at Dartmouth explained to me the advantages of controlling his own computing, free from the juggling act that is inevitable with a central computer that serves the entire university. Recently, for my research at Cornell, I was in a similar position. Our group had its own large computer cluster and we made all the decisions about how to use it.

Computer centers were also worried about the costs of supporting large numbers of different types of computers. Undoubtedly such proliferation did cause difficulties and universities tried various approaches to limit the variety. When I arrived at Carnegie Mellon there were 101 different makes of computer on campus. One of my first acts was to abolish an ineffective policy where I, as vice president for computing, had to approve all purchases of departmental computers. In later years we developed a short list of personal computers that we supported centrally. Most members of the university bought these types of computer, relying on us for support, but there were always a few mavericks who chose differently. These mavericks were invaluable in evaluating new options.

At the universities that I know best, Dartmouth, Carnegie Mellon, and Cornell, the various organizations eventually learned how to work together and support each other, but even today there remains an underlying tension between centralization and decentralization at almost every research university.

The Computer Industry

Motives and motivation

The interactions between universities and computing companies were beneficial for both. The universities' basic motive was straightforward: money. As a politician once said to me, "I know why you are here. Universities always want something for nothing." Over the years we bought computers at deep discount from Honeywell, Digital, Prime, IBM, Apple, Sun, Dell, and many more. We received gifts of millions of dollars of equipment, and cash grants of millions of dollars for specific projects.

What have the companies received in return? First there are sales. Universities are a major customer. IBM, Digital, and Apple were among the companies that cultivated this market with deep discounts and donations of equipment. In the early days of personal computers, IBM gave universities huge grants of equipment in the hope that the universities would standardize on the IBM PC and write leading-edge software for it. But sales were not the only motivation. Many people in the computing industry are genuinely interested in education and are strong believers in technology for education.

Universities were early adopters of many types of computers and the first organizations to take networking seriously. As such, we were important for companies trying to introduce novel products. When Sun was a start-up, Carnegie Mellon was its biggest customer. Today it is hard to remember that the Apple Macintosh was a commercial flop when introduced in 1984. Without orders from a small number of universities, it would probably have died within the first year.

An apple from Apple

This Steuben Apple was given to the president of Dartmouth in 1984. He passed it on to me. I think that he was embarrassed to be buying computers from Apple, which was seen as a slightly improper upstart company.

Photograph by William Arms

Companies valued our insights. By briefing us on new ideas before they became products, companies could get early feedback. They presented their plans and we were not shy in telling them what we needed. When Bell Atlantic (now Verizon) released their first cell phone service, it was a multi-million-dollar gamble. The vice president in charge told me that the only market research that he trusted was from an experimental installation at Carnegie Mellon. When NCR developed Wi-Fi networking, they came to us for an alpha test. When Steve Jobs showed his NeXT computer to a group of university leaders, we told him that universities would not buy a Unix workstation without the X window manager. He did not like it, but we needed it, and he believed us.

The companies also supported university research. Digital was particularly well organized in supporting external research. Most of their support was equipment grants, but for the right project they were willing to provide large sums of money. Intellectual property can be a problem with corporate research, but Digital allowed universities to keep the rights. Their benefits came from exchanges of ideas with researchers, and from leading-edge software being developed on their systems. One of my major disappointments when at Carnegie Mellon was a large grant from Apple to port Unix and the entire Andrew environment to the Macintosh II, and to integrate it with the Macintosh user interface. This was brilliantly done, but when we came to deliver it, Apple had disbanded the engineering group that had sponsored us and the work was wasted.

Finally, companies are always looking for good recruits. Ph.D. and master students from the best universities are in enormous demand. Later in my career, at Cornell, I felt as though I was running a farm system for the Internet industry, teaching upper-level courses on information retrieval and software engineering, with teams of students doing independent research on an Hadoop cluster. Even in the depth of the recent recession companies were hungry for these graduates. Many times each year a student will ask my advice on which job offer to accept. One Carnegie Mellon student had to choose between a faculty position at Stanford and a central position at a fascinating start-up. He chose the start-up.

I think that the corporate people enjoyed visiting the universities. They would often ask to meet students, see demonstrations of research, or attend events on campus. A colleague at Penn State organized his requests for donations around the football season. Carnegie Mellon was at one time a part-owner of the Pittsburgh Pirates baseball team and we would entertain visitors in the owners' box. A visitor from Stanford University even went back to California and persuaded Apple to take a box at Candlestick Park.

Exchanges of information

Visits to the corporations were enjoyable and productive. We would sign a non-disclosure agreement, the company's engineers would brief us on their plans, and we would provide our feedback. Our local salesmen came with us and often learned facts about their companies' plans that were not usually known to the sales force. Each year a group of us from Carnegie Mellon went to California for a week, spending a day each at Sun, Apple, Hewlett-Packard, NeXT, Adobe, and other companies. Other visits were to Digital's research labs outside Boston and the various IBM locations.

Several companies had formal university groups. The first meeting of the Apple University Consortium was a very special occasion. It was in San Jose in December 1983, before the official announcement of the Macintosh. We were given early information about future products and shown the famous 1984 Super Bowl commercial. The consortium later grew too large and lost its effectiveness, but IBM's university advisory group was always small. Each year it concentrated on a single topic. One year the topic was networking. IBM was throwing its main effort into OSI networking, while also contributing large sums to build the NSFnet, which became the backbone for the Internet. The IBM vice president for networking spent much of the day failing to convince us of the virtues of OSI and not understanding why we preferred the Internet.

Only a few universities had this privileged access to the companies. To reach a wider audience, the companies had booths at the EDUCOM conference and hospitality suites where we could talk informally. These grew increasingly lavish until, after consulting with his lawyer, the IBM head of university marketing sent a letter to his competitors, "Let's not have food fights. We should compete with our products and services, not with the lavishness of our hospitality." In contrast to this sensible attitude, I attended a couple of corporate seminars and was appalled by the lack of serious content and the extravagant hospitality. The worst was a so-called seminar by Bell Atlantic. It was a disgrace: minimal content, but expensive food, drink, golf, and sea fishing.

Salesmen

While at English Electric - Leo I had a first-hand look at how computer systems are sold and learned some useful lessons. One lesson was never to trust demonstrations. Multiprogramming was a major selling point for the System 4 computers but the operating system was incomplete. The company regularly ran demonstrations that showed several programs running at the same time. Card readers chattered away, magnetic tapes whirled, and the line printers churned out paper, but these demonstrations were faked. Actually these three programs were the only ones that could be run together.

A second lesson was that salesmen offer the configuration that will get the order, not the one that will do the job. We were asked to bid on a system for a company that ran off-track betting. They had difficulty finding people to operate their telephone center on public holidays, when there are many horse races. Consultants had recommended a computer system and provided a price estimate. Our salesman configured a bid based on that price and I was asked to write a section about performance for the proposal. My calculations showed that the configuration was hopelessly too small, but the salesman submitted the bid anyhow. Fortunately the contract went to a competitor, Control Data Corporation, who ran into all sorts of problems. Years later, another consultant solved the staffing problem by the simple expedient of moving the telephone center from central London to an area with high unemployment.

Years later, as a potential customer, we received a bid from Hewlett-Packard that revolved around a suite of programs for business data processing. Not only was the memory in the configuration too little for the programs that were offered, the machine that was quoted could not be extended. All this information was carefully buried in the tender.

The most important lesson was always to know how the salesmen were rewarded. At Dartmouth we used Honeywell mainframes. Honeywell may have had a price list, but they paid their salesmen by the dollar amount that they booked. To buy a computer we first agreed on the price and then negotiated on what would be delivered. On one occasion, a salesman booked an option to buy a second machine and Honeywell recorded it as an order. Several years later a truck from Arkansas arrived at the Dartmouth computing center in New Hampshire with a large mainframe computer. Fortunately the operator on duty refused delivery. On another occasion Honeywell delivered the wrong central processor. On a happier note, we completed a complex negotiation with Honeywell just before Christmas one year. As soon as the contract was signed, but not before, the salesman delivered a Christmas present. It was a clock. Like many of Honeywell's products, it was made in Japan.

The best time to buy equipment was the week before the company closed its books, as they were always trying to bolster their sales figures. As a computing director the most fundamental rule is that you stay within budget, but budgets are developed a long time before the funds are spent. Private universities have the flexibility to take advantage of opportunities. Twice at Dartmouth, when everybody else had left for the Christmas break, I sat in President Kemeny's office asking for extra funds to take advantage of a year-end offer. At most state universities every line in the budget is fixed and the universities do not have this flexibility.

If the Digital salesmen met their sales target they received rewards. One year our salesman came to me with a hard luck story. He was one computer short of having met his target for five years in a row. Please would I order a VAX system from him? I could cancel the order a week later. Naturally I refused, but he was an excellent salesman and I called his manager. He got his reward. As I remember it was a trip to Jamaica with his wife.

Aggressive marketing

The marketing strategy perfected by IBM for selling mainframe computers was highly centralized. The salesmen cultivated relationships with the computing directors and the customers' higher executives. In many organizations the purchasing of computers remained centralized, even for minicomputers and personal computers, and the IBM method of selling directly to senior executives continued to be successful. In universities, however, as computing became decentralized, purchasing decisions were made by departments. The minicomputer salesmen understood this and developed relationships with the departments. As Barbara Morgan of the University of California at Berkeley quipped, "When the IBM salesman visits me he asks, 'What is happening at Berkeley?' When the Digital salesman visits me, I ask him, 'What is happening at Berkeley?' "

IBM had a reputation for bypassing the computing professionals and selling directly to the president of an organization. This was called an "end run". I could not understand why a chief executive would listen to somebody on a golf course rather than to the experts in his own organization, but it was very effective. When I worked for a strong president, such as John Kemeny at Dartmouth or Dick Cyert at Carnegie Mellon, this was no problem, but their successors were weaker. Kemeny's successor at Dartmouth had an IBM vice president as a buddy who caused enormous difficulties. He appeared to be ashamed that we did business with less prestigious companies such as Digital and Apple. Cyert's successor cost Carnegie Mellon a large sum of money by throwing out a carefully negotiated contract with NEC for a telephone switch because of an end run by the president of Bell Atlantic. He was a Carnegie Mellon alumnus who hinted that there might be grants to the university.

IBM and Digital were aggressive but they were good companies to do business with. The telecommunications companies were frequently dishonorable. The computing companies honored the spirit of an agreement, even

if it was only a manager's letter, but the telecommunications companies were always trying to wriggle out of commitments.

Planning for Personal Computers

Sources, reliable and unreliable

Between 1980 and 1995, networks of personal computers replaced timesharing as the core of academic computing. For much of this period, academic computing and the commercial mainstream continued to follow separate paths, but they eventually converged in the 1990s.

Sources of information about this transformation are scattered and unreliable. Much of what was written at the time described what was planned, which was often not what was achieved. Many of the newspaper articles were university publicity laden with euphoria. *The New York Times* had a correspondent who was well known for presenting his personal biases in the guise of factual reporting. Perhaps the most reliable source would be the notes of Judith Turner from the *Chronicle of Higher Education*, but she has wisely kept her notes private.

Much of what has been written subsequently is pure revisionist history. Individuals and organizations describe what they wish had happened. Wikipedia is frequently a good source of technical details, but the contextual information is often tendentious. In practice we went through a decade of hyperbole, resistance to change, blind alleys, failed plans, and spectacular achievements.

Three books published by EDUCOM are a good contemporary source. EDUCOM was the university computing society. Its publications, annual conference, and corporate associates' program were an important exchange of information. I was a member of the EDUCOM board for most of the 1980s and its chair for six years. Under two outstanding presidents, Jack McCredie and Ken King, EDUCOM was a major conduit for contacts with corporations and the government.

One day in fall 1980 Jack McCredie and I were walking across the Stanford campus. He mentioned that he was planning a monograph on university computing. Ten universities would each write a chapter about their strategic planning. I casually agreed to be one of the authors without admitting that at Dartmouth we were so busy building our computer system that we did not do any formal planning. This was the first of the three books. My wife edited the other two, which were about campus networking and libraries.

Two of Ken King's creations were the Coalition for Networked Information, which brought librarians and computing center directors together, and a networking task force that lobbied for higher education in the development of national networks. Al Gore is rightly ridiculed for his claim to have invented the Internet, but he was the first high-level politician to recognize its potential and a great supporter of these efforts.

Because I served on numerous committees, I often saw planning papers from other universities. About 1980, Stanford wrote a particularly insightful set of papers on topics such as networking, office automation, libraries, and so on, but, for me, the most illuminating was the *Preliminary Report of the Task Force for the Future of Computing at Carnegie Mellon*.

Managing the first personal computers

Personal computers posed a dilemma for universities and their computing centers. After years of struggle, we had learned how to manage timeshared computers. A few years earlier, some computing directors had resisted the purchase of minicomputers by departments, but eventually they accepted that minicomputers were frequently cost-effective. Now they had to face a wave of incompatible personal computers that were often purchased by people with little knowledge of computing. The total expenditure on computing was going up steadi-

ly and it was not clear what the university was gaining. For a generation who believed that every computer cycle was precious, it was painful to see computers sitting on desks to be used for only a few hours every day. Some computing directors thought that their mission was to protect the university from this wasteful proliferation.

Gradually people began to realize that personal computers were more than a fad. Admittedly they could not do many of the tasks that the larger machines did well, but every year they became more powerful and the application programs grew better. For example, in summer 1982, a student and I developed a planning model for hardware purchases using Visicalc on an Apple II. The time and effort were a fraction of what we would have taken on a timeshared computer. Above all, people began to appreciate the benefits of having their own machines. I used to compare personal computers to private cars. It might be cheaper and safer for everybody to travel by bus, but individuals pay for the convenience of having their own cars.

The showpiece projects

At the beginning of the 1980s, the academic world was excited by the launching of ambitious computing projects at several universities. The most prominent was the Andrew project at Carnegie Mellon University, which included a massive grant from IBM. Never to be outdone, MIT soon afterwards announced the Athena project, jointly with IBM and Digital. In the same year, 1982, Drexel University announced that every freshman would be expected to have a personal computer, and some time later they put out a press release stating that they had chosen an unnamed computer from Apple (which they described as being supplied with a "house", a typo for "mouse"). This computer was later known as the Macintosh.

Several other universities, notably Brown University and Franklin and Marshall College, had important initiatives, some universities passively accepted the arrival of personal computers on campus, and some universities actively discouraged the proliferation. Where there was no campus-wide project, departments or schools, such as engineering or business, often built their own local networks. Harvard is an example of a university that deliberately held back, waited to see what consensus emerged, and then purchased a very fine network, but Harvard is a rich university and could afford to wait. The pioneers endured a great deal of hassle, but received major grants of equipment and money. Brian Hawkins, a force behind the Drexel program, once observed that the universities that took the lead were usually the universities with strong academic leadership.

Whether students should be required to own personal computers formed a lively topic throughout the decade. At Dartmouth we standardized on the Macintosh in 1984 and at our urging, more than 75 percent of freshmen bought them, but not until years later were students required to have computers. Carnegie Mellon was more typical. Although the university had a huge project with IBM, no explicit requirements were made to buy computers. The campus store sold personal computers from both IBM and Apple, as well as workstations from several other manufacturers, with deep discounts for hardware and software.

Organization for distributed computing

The story of these large projects is incomplete without a discussion of the struggles that the universities went through to fit them into their organizations. Universities always have great difficulty in coming to a consensus, particularly when resources have to be reallocated, and these projects were no exception.

In a talk in 1985, Steven Lerman, one of the joint heads of the Athena project at MIT, described a problem which he called "the management of expectations." Projects such as Andrew or Athena need great initial enthusiasm to marshal the resources that they require. Then follows a long gestation period before any signs of progress are seen, and the first results are a pale reflection of the dream with which the projects began. Carnegie Mellon is an extraordinarily entrepreneurial university in its willingness to tackle new ideas, but the university is not rich. The growth in computing demanded resources of money and space, which were tough to find, at exactly

the same time that the Andrew project was struggling to live up to its expectations. Fortunately, President Cyert was personally committed to the project and the resources were found.

As the importance of computing grew, job titles were inflated. At the Open University in the 1970s, the head of academic computing was simply the manager of student computing. At Dartmouth my title changed from director of computing services to vice provost and at Carnegie Mellon it was vice president. These distinctions seemed important at the time.

Whatever their titles, computing directors were in a difficult situation during the transition to decentralized computing. Some saw their duty as protecting the central computing budget from the inefficiencies of departmental and personal computing. Others allowed the central service to atrophy while they invested in networks and decentralized computing. No university had the resources to do both really well. During the transition almost every university changed its computing director. Sometimes the university forced them out. Others left for better opportunities or because of frustration with the university's administration. I had been drawn to Dartmouth and Carnegie Mellon because of two outstanding presidents, John Kemeny and Richard Cyert. Working for their successors was no longer enjoyable and I moved on.

Until about 1980, most computers were purchased with central funds and a handful of people made all the decisions. Only in rare instances were computers seen as personal equipment. A decade later the position had been reversed. In 1990, the annual expenditures on computing equipment at Carnegie Mellon were about $12 million, with only $2 million coming from central funds. Purchases made with personal funds through the computer store were over $3 million, and the rest came from departmental funds or research grants.

People who spend their own money naturally expect to choose their own equipment. Yet, everybody benefits from a certain level of campus-wide coordination. The organizational challenge was to reconcile independence in decision making with sufficient standardization to create a coherent campus environment. At Dartmouth, the campus was dominated by Macintoshes, but there were still large numbers of other computers. At Carnegie Mellon, although no formal commitment was made, the expectation had been that the Andrew project would lead to a largely homogeneous environment dominated by IBM. This did not happen. A tour of the campus in 1990 would have found Apple Macintosh and IBM personal computers everywhere, with hundreds of workstations made by IBM, Sun, Digital, Hewlett-Packard, and NeXT. These computers had been selected by many different people at different times with differing needs and resources. Each selected the equipment which best fitted current needs. Yet, at every university, a certain level of standardization emerged because of the difficulty of supporting too wide a range of equipment. The computing environment was so complex that only a few types of computer could be well integrated into it. Most people bought those types of computers.

Gradually universities came to realize that they needed somebody who would oversee the computing strategy for the entire university. This person was not necessarily the person who managed the central computers and did not have to be in charge of both academic and administrative computing. At Carnegie Mellon I had responsibility for academic computing and infrastructure, such as networking, but not for administrative computing. This division between academic and administrative worked well. I am surprised that it is not more common.

At Dartmouth, the Kiewit Network and the Macintosh project were creations of the computer center and there was never any discussion of creating a new organization. Andrew at Carnegie Mellon and Athena at MIT both began as separate organizations outside the existing computer centers, but thereafter followed different paths. Athena remains a separate unit, but at Carnegie Mellon I transferred the responsibility for deploying Andrew to the old computer center. This required a painful reorganization, but the organizational framework has survived.

The institutional structures for academic computing that have emerged at most universities usually parallel the organization of the rest of the university. The organization that I see today at Cornell is typical of large research

universities, partly centralized and partly decentralized. It combines aspects of both extremes: entrepreneurial faculty with powerful and independent deans, and yet a fairly strong computing center that provides shared services such as networking and email to the whole university. This central organization is also responsible for the university's administrative data processing

Chapter 4

Networks

Installing a network

This photograph from the University of Michigan shows the unglamorous side of networking, running cables within and between buildings. In the 1980s, the labor costs of wiring old buildings was more than $200 per outlet.

Photograph by Geotech, Inc.

Campus Networks

The starting point

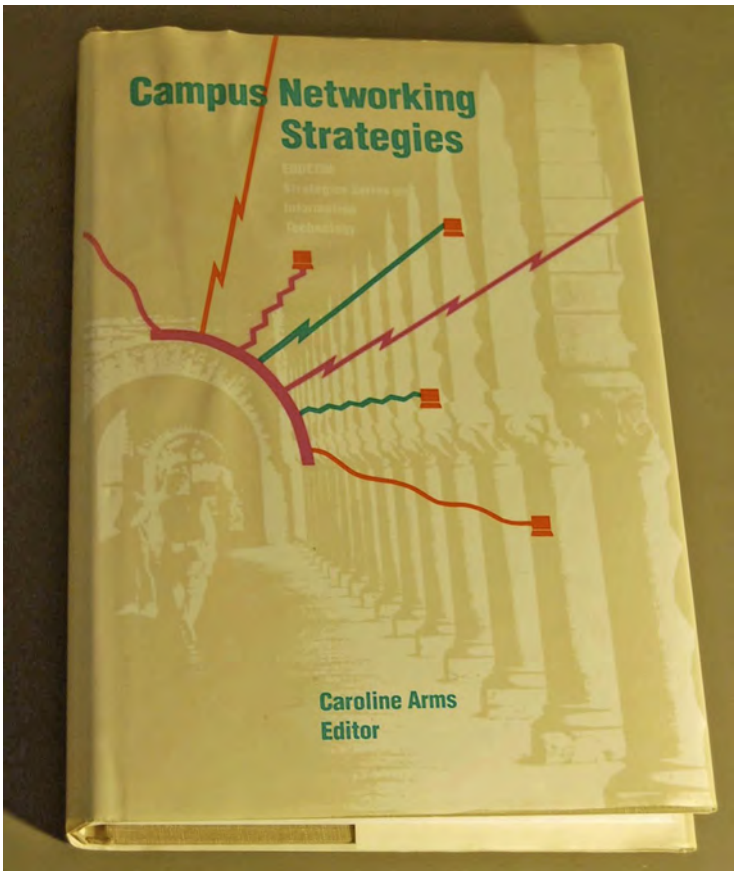
In October 1977, while on my way to be interviewed at Dartmouth, I spent some time at Cornell and contacted the computer center to ask if somebody could discuss Cornell's computing with me. By a spectacular piece of fortune my guide was Douglas Van Houweling, who later became one of the leaders of networked computing. As we walked around the campus on a beautiful fall day he described the need for computer networks in universities.

The concept of networking was not new. Universities already connected large numbers of terminals to central computers. A common architecture was to run groups of asynchronous lines to a terminal concentrator and a higher-speed, synchronous line from there to the central computer. The lines might be hard-wired or dial-up telephone lines. IBM had an architecture known as SNA, which provided hierarchical ways to connect terminals and minicomputers to a central computer, and Digital had built its DECnet protocols into the VAX and DEC-20 operating systems. The ARPANET was well established among a privileged group of researchers, and Xerox had begun the experiments that eventually led to the dominance of Ethernet.

Doug's insight was to realize that these activities were not isolated. At Cornell, departmental computing centers were springing up independently from the main computing center. Very soon they would have to be connected together. Universities would need to have networks that covered the entire campus and connected every type of computing device. They had to use protocols that were not tied to any specific vendor.

Campus strategies

In the EDUCOM book on campus networking, ten universities, large and small, reported on their mixtures of terminal concentrators, SNA, DECnet, Wangnet, X.25, AppleTalk and many more. In the larger universities, the mixture of



Campus networking strategies

The photograph shows the second of the three EDUCOM books about campus strategies. The book about networking was particularly well-timed.

Photograph by William Arms

networks was proving a nightmare. Well-funded areas of the university might have several disjointed networks, while others had none. Providing gateways between networks sounded fine in theory but never seemed to work in practice. Network management was rudimentary and trouble-shooting was expensive. Yet when the chapters were assembled it emerged that, with one exception, the universities had independently decided on the same solution. The strategy was to create a homogenous campus network and to use the Internet protocols for the backbone.

I was responsible for two campus networks. The similarities and differences between them reflect the different priorities and resources of the two universities. At Dartmouth we built a campus network that emphasized moderate performance at a low cost. Because we started early, the problem of incompatible protocols was never as severe as at other universities, and the campus-wide adoption of Macintosh computers allowed us to use AppleTalk as a default protocol. Later, when I went to Carnegie Mellon we built a much more powerful network. The university recognized the strategic value of the campus network and provided a very substantial budget. With generous support from IBM, and with collaboration among many groups at the university, we built a TCP/IP network and connected a wide variety of computers to it. Both networks proved successful and their successors are still in operation today.

Topology and wiring

Networks are expensive. In many universities, although the computing directors and leading faculty members recognized the need for a network, it took years to persuade the financial administrators that universities needed to build them. Even at Carnegie Mellon, part of the justification for wiring the campus was the promise of a much improved telephone system.

A crucial decision was the topology of the campus wiring, within buildings and between them. With several options to choose from, a university could easily build an expensive network that went out-of-date fast. Initially the leading candidates were variants of a bus. Early Ethernet ran on a 50-ohm coaxial cable. This would be snaked through a building and individual devices clamped onto it. IBM had a token ring architecture. AppleTalk used a variant of a bus in which devices were daisy chained together using special cables. An Ethernet bus or a token ring were possibilities for the backbone between buildings, but another candidate was to use 75-ohm video cable with broadband modems.

A bus appears appealing, but has some serious difficulties in practice, and the topology that emerged was a star shape. Within buildings, communication lines were run back to a building hub, and from there lines were led to a central hub. Ethernet and other protocols were modified so that they could run over these lines. The star-shaped configuration has many practical advantages. If part of the network becomes overloaded, extra capacity can be installed by upgrading the equipment in selected building hubs. Almost all trouble-shooting can be done at the hubs. On a bus, when there is a problem, it may be necessary to visit every device.

Eventually star-shaped Ethernet over copper wires became the standard at almost every university and the Internet protocols squeezed out the vendor networks, but for many years we had to support a variety of interfaces and protocols. A later section describes the detailed choices made at Dartmouth and Carnegie Mellon.

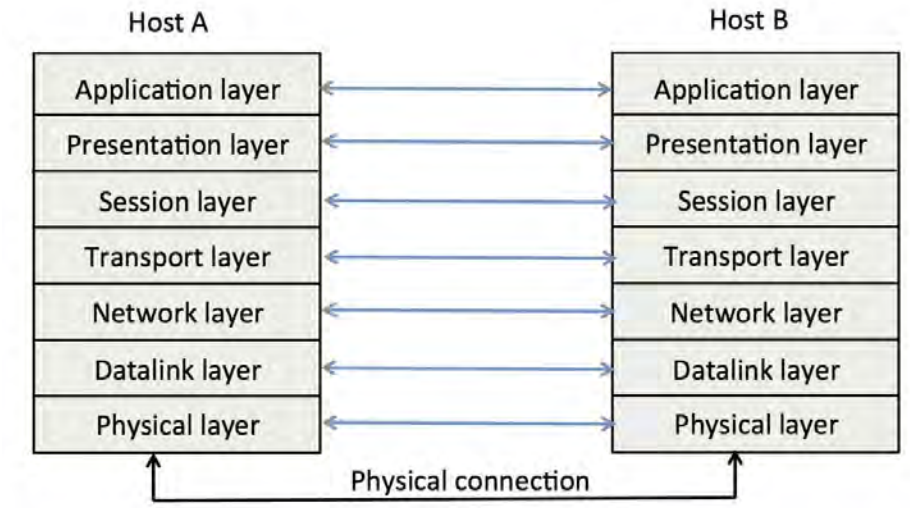
Running wires within buildings was a major expense and the choices were not easy. At Dartmouth, we made a typically pragmatic decision. We decided to use cheap copper wire within buildings, which was easy to install, expecting that it would have a limited life. Fortunately, when the university became interested in our efforts, the trustee who was asked to look at the network was an engineer and he agreed with our approach. However, ten years later, when the time came to renew the wiring, I have heard that this rationale had been forgotten by the new administration. A few years later, Carnegie Mellon made a different decision. The network that we installed there used IBM's cabling system. This was designed to provide a choice of high-quality networking options for many years into the future, for both data and voice communication. It was expensive to install but provided excellent service.

National Networks

The Open Systems Interconnection (OSI) model

By the early 1980s the benefits of networking had become generally recognized, but building the networks proved to be challenging. The developments fell into two categories: the campus networks and the networks between universities. Together they developed into the Internet that we know today. If you read the conventional history of the Internet, you could easily believe that the Internet protocols were so superior that it was inevitable that they would be used for both a national network and for campus networking. At the time, however, it was far from inevitable.

Few people remember that the telecommunications companies and the computing industry fought for years to kill the Internet and its TCP/IP family of protocols. They recognized the potential market and the need for a shared set of protocols, but they wanted to control everything. They embarked on an intense lobbying effort to suppress any competition, using every political resource to stop alternatives, and were particularly outspoken about wasting government money on TCP/IP developments. The telecommunications companies, who have made such enormous profits from the Internet, were the most opposed to its development.



The OSI seven-layer protocol model

Diagram by William Arms

The diagram above shows a popular model that provided a framework for discussing networks. The model divides communication protocols into seven logical layers. Each layer serves the layer above it and is served by the layer below. In this model, Ethernet combines the physical and datalink layers, IP and X.25 are network protocols, TCP is a transport protocol, and FTP and Z39.50 are application protocols.

Rather than adopt the Internet protocols, the industry set out on an ambitious program known as the Open Systems Interconnection (OSI). OSI was supported by all the major companies. Technically it had two main parts: standardization and creating commercial products. IBM alone spent hundreds of millions of dollars. It seemed inevitable that the national and international networks would be built to these standards. Parts of OSI, such as X.25, were successful, but as a whole it suffered from over-complexity and eventually collapsed under its own weight.

Meanwhile universities had difficult decisions to make about what protocols to use on campus and for academic networks. For years, they wrote that their networking plan was to use TCP/IP "until OSI becomes available." Eventually they dropped the final clause. How did this happen?

The first national networks

When universities began to recognize the potential of a national network there were several possibilities. The dominant network between universities was Bitnet. This was a store-and-forward network that ran between IBM mainframes. It used software that IBM had developed for its own internal network. Bitnet provided the first widespread email service between universities and was used for most bulletin boards (called LISTSERVs) until the web became established in the 1990s. As late as 1986, there was no way to send an email message between Dartmouth and Carnegie Mellon except over Bitnet.

The first commercial network was Telenet, which offered X.25 services for a monthly fee. CSnet was an X.25 network developed by those computer science departments who were excluded from the ARPANET. At Dartmouth, we used Telenet for remote terminal access and CSnet for email.

The ARPANET was a higher-speed network developed by the Department of Defense's Advanced Research Project Agency (ARPA). The TCP/IP protocols were developed for the ARPANET. A select group of universities were members of ARPANET, but its use was restricted to people who were doing ARPA-related work. At Carnegie Mellon we were always careful to observe this restriction until I learned that MIT ignored it. They considered that everybody on their campus was doing ARPA-related work.

The success of TCP/IP

One reason that OSI failed was that it attempted to standardize everything. With little operational experience to guide them, the standardization groups added more and more features. The early Internet was much more pragmatic. Its motto was "rough consensus and running code." TCP/IP specified the network and transport protocols, but made no attempt to define the underlying network technology, while protocols for applications such as email were not standardized until there was practical experience with running code.

A key to the success of TCP/IP was the decision to include the protocols in the Berkeley Software Distribution of Unix (BSD). This was also funded by ARPA. BSD 4.2, released in 1983, had an "open source" version of the full family of protocols. This code, which was soon ported to many other operating systems, was the basis for the first generation of the Internet.

In hindsight, the benefits to university computing are obvious. We could use the same protocols on campus and off-campus without the need for complex and inefficient gateways. Because all the versions were based on the same implementation, there were few compatibility problems. TCP/IP came from the computer science research community, which has a tradition of shared problem solving, and it was backed by a well-funded government agency with strong interests in its success.

The NSFnet

The crucial event was the 1985 decision by the National Science Foundation to establish five supercomputer centers for academic research, to connect them with a high-speed network, and to base it on the Internet protocols. Again this was not inevitable. The NSF's newly formed networking group had a hard fight to stop the traditional disciplines from taking their money, and to resist lobbying by commercial interests who were opposed to something that they did not control. It helped that the NSF's effort was led by an Irishman who could charm anybody. The NSFnet backbone was built by a consortium put together by Doug Van Houweling at the University of Michigan with generous support from IBM and MCI. IBM provided the routers and MCI supplied the T-1 connections, which at 1.5 Mbits per second were far beyond the speed of any previous data networks.

The NSF also funded an enlightened program to extend the NSFnet by creating regional networks. Our experience at Carnegie Mellon was typical. The computing directors of the major Pennsylvania universities formed

a team led by Gary Auguston of Penn State. His design criteria were, "TCP/IP, T-1, free." With support from Bell Atlantic, we met with a senior member of the governor's staff. His first question was, "How will this get the governor re-elected?" His second was, "How can I help?" We then collectively applied to the NSF for a substantial grant. This grant supported the Pennsylvania regional network for long enough to become self-sustaining.

Finally, the NSF successfully transferred management of the network to the commercial sector in 1995. This was a shock to the universities, as henceforward we had to pay for our networking.

Case Studies: Dartmouth and Carnegie Mellon

The Kiewit Network at Dartmouth

Dartmouth can claim to have had the first complete campus network, the Kiewit Network. The architect of the network was Stan Dunten, who had earlier been the developer of the communications for the Dartmouth Time Sharing System (DTSS) and before that a major contributor to MIT's Multics. His concept was to create a network of minicomputers dedicated to communications, interconnected by high-speed links. He called them nodes. The nodes would be used as terminal concentrators, as routers to connect computers together, or as protocol converters. Earlier, when the university had been laying coaxial cable for television, he had arranged for a second cable to be pulled for networking. This cable was never actually used for its purpose but its existence was an important stimulus.

The first two nodes were the Honeywell 716 minicomputers that were the front ends for the central DTSS computer. They acted as terminal concentrators and provided front-end services such as buffering lines of input before sending them to the central computer.

During the 1970s, Dartmouth sold time on the DTSS computer to colleges that were too small to have their own computers. The first extension of the network was to support these remote users. The front-end software was ported to Prime 200 minicomputers, which had the same instruction set as the 716s. They were deployed as terminal concentrators at several colleges, with the first being installed at the US Coast Guard Academy in 1977. These nodes were linked to Dartmouth over leased telephone lines using HDLC, a synchronous data link protocol.

In fall 1978 the decision was made to build a new set of nodes, using 16-bit minicomputers from New England Digital. Eventually about 100 of the New England Digital minicomputers were deployed on and around campus. The original development team was Stan Dunten and David Pearson, who were later joined by Rich Brown. Each node could support 56 asynchronous terminal ports at 9,600 bits/second, or 12 HDLC synchronous interface at speeds up to 56Kbits/sec. An important feature was a mechanism to reload a node automatically from a master node. After a failure, a dump of the node's memory was stored on DTSS and the master node sent out a new version of the network software. The master node also performed routine monitoring and individual nodes could be accessed remotely for trouble-shooting.

The first new service was to allow any terminal or personal computer emulating a terminal to connect to any computer on the network. In 1981 the university library connected its first experimental online catalog to the network, which was immediately accessible from every terminal at Dartmouth. The production version of the catalog ran on a Digital VAX computer. Other computers that were soon attached to the network included a variety of minicomputers, mainly from Prime and Digital. The first protocol gateway provided support for X.25, which was used for off-campus connections to the commercial Telenet service and later to the computer science network, CSnet.

Wiring the buildings

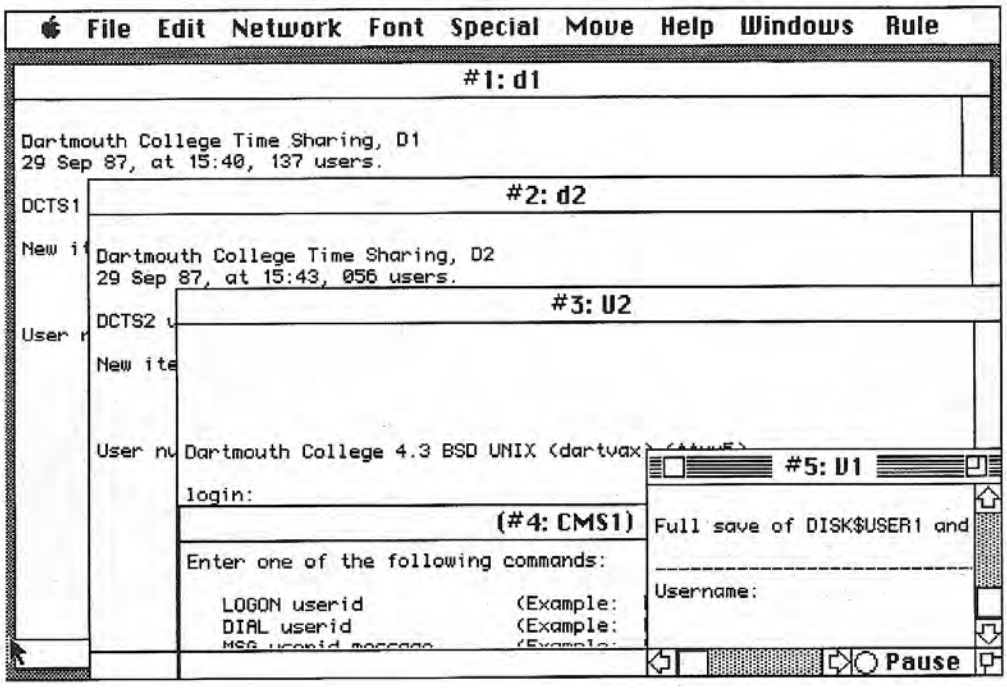
Before the Kiewit Network was built, terminals were connected to DTSS over standard telephone lines using acoustic couplers. Dedicated hard-wires were run where the terminals were close to a front-end computer, such as in the computer center or at the Coast Guard Academy. As the New England Digital nodes were deployed, the number of hardwired terminals increased and as funds became available the campus buildings were wired. The funding model was simple. Hundreds of people around the university were accustomed to paying $16 per month for a 300 bits/second acoustic coupler and a special telephone line. As the hardwired network was gradually installed, users paid $15 per month for a hardwired connection at 2,400 bits/second (later 9,600 bits/second) and the funds were used for the next stage of expansion.

The Dartmouth campus is crisscrossed by roads and in those days the telephone company had a monopoly on transmission lines that crossed the roads. For many years the high prices charged by the telephone company forced the network to use low-speed links between the nodes. At a time when Carnegie Mellon, with no roads across its central campus, was laying its own fiber optics cables, many of Dartmouth's links were only 19.2 Kbits/second. Yet even with these constraints the performance was excellent for the primary purpose of linking terminals and personal computers to central computers.

The Macintosh network

In 1983, Dartmouth decided to urge all freshmen to buy Macintosh computers. This required a rapid extension of the network to support Macintoshes and to wire the dormitories. Before this expansion, the university's central administration had essentially ignored the development of the Kiewit Network. It began as an internal project of the computer center, appeared on no public plan, and never had a formal budget. Now, the university recognized its importance and asked the Pew Foundation for the funds to extend the network to the dormitories. With a generous grant from the foundation, the network was fully deployed by fall 1984, with hardwired ports in every room including the dormitories. The nodes were configured so that each hardwired connection could be used for either AppleTalk or as an asynchronous terminal port. By default the dormitory ports were set to AppleTalk, one connection per student, but could be changed on request.

Originally communication between the nodes used the unique DTSS protocol, but it became clear that something more modern was needed. TCP/IP was rejected for a variety of reasons. The overhead was too high for the slower network links (some off-campus nodes were 2,400 bits/second) and there was no TCP/IP support for the major computers on campus. Before the Internet Domain Name Service was introduced in 1983, whenever a machine was added to a TCP/IP network, a system administrator had to update all the name tables. Dartmouth was considering a stripped down version of TCP/IP when, in spring 1984, Apple showed us the draft protocols for what became AppleTalk. These protocols were very similar to our own design, being packet switched but simpler than TCP/IP. The networking team decided to adopt Apple's protocols and persuaded them to make some changes, notably in the size of the address field. These protocols were adopted for the internal links of the network and the Kiewit Network became a large AppleTalk network in 1984, with the nodes acting as gateways for other protocols.



Using a Macintosh as a terminal

This photograph shows five connections from a Macintosh to timeshared computers. Terminal emulation was an important service during the transition from timesharing to personal computing.

Screen image by Rich Brown

There was perilously little time to complete the changeover before the freshmen arrived in fall 1984. It would not have been possible without special support from Apple. The company helped in two vital ways. The first began as a courtesy call by Martin Haeberli who had just completed MacTerminal, the terminal emulator for the Macintosh. For a vital few days he became a member of the Kiewit team. The second was a personal intervention from Steve Jobs. The isolation transformers that protect AppleTalk devices from power surges were in very short supply. Jobs made sure that we were supplied, even before the Apple developers, rather than risk deploying a thousand machines without protection from lighting strikes. AppleTalk was officially released in 1985, six months after it was in production at Dartmouth.

The adoption of AppleTalk for the campus network had a benefit that nobody fully anticipated. Apple rapidly developed a fine array of network services, such as file servers and shared printers. As these were released they immediately became available to the entire campus. Thus the Macintosh computers came to have three functions: free-standing personal computers, terminals to timeshared computers including the library, and members of a rich distributed environment.

Later developments

By building its own nodes, Dartmouth was able to offer the campus a convenient, low-cost network before commercial components became available. It served its purpose well and was able to expand incrementally as the demand grew for extensions.

Even as the AppleTalk network was being deployed, other systems were growing in importance. Scientists and engineers acquired Unix workstations. The business school was using IBM Personal Computers. Ethernet and TCP/IP became established as the universal networking standards and eventually both Dartmouth and Apple moved away from AppleTalk. During the 1990s Dartmouth steadily replaced and upgraded all the original components. 1n 1991 the connections between the building were upgraded to 10 Mbits/second and today they are fiber optic links at 10 Gbits/second. The building wiring was steadily replaced and by 1998 all the network outlets had been converted to Ethernet. In 2001, Dartmouth was the first campus to install a comprehensive wireless network. The last New England Digital nodes were retired many years ago, but the basic architecture that Stan Dunten designed in 1978 has stood the test of time.

The Andrew Network at Carnegie Mellon

As part of the Andrew project at Carnegie Mellon, a state-of-the-art campus network was deployed between 1985 and 1988. By designing this network somewhat later than Dartmouth, Carnegie Mellon was able to build a higher-performance network, but had to accommodate the independent networks that had been installed on campus. The most important of these were a large terminal network using commercial switches, several networks running DECnet over Ethernet, and the Computer Science department's large TCP/IP network. Early in the decade, a working group convened by Howard Wactlar persuaded the university to accept the TCP/IP protocols for interdepartmental communications, but the departmental networks were much too important to be abandoned.

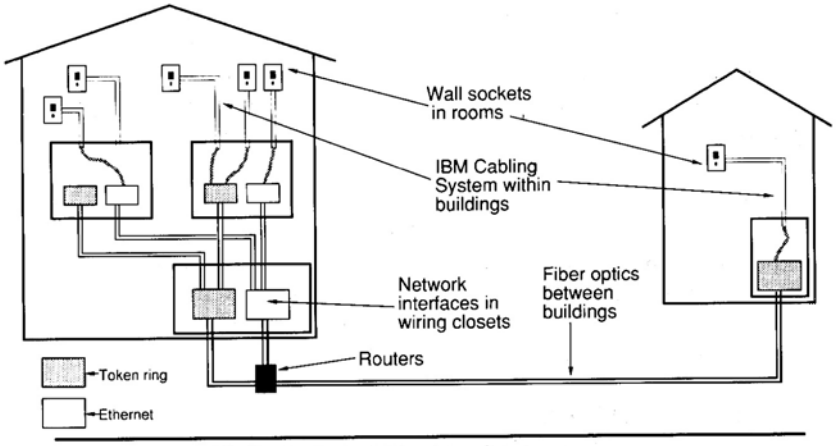
Early Ethernet posed problems for a campus network. The network management tools were primitive and large Ethernets had a reputation for failing unpredictably when components from different manufacturers were used together. To manage the load, the typical configuration was to divide a large Ethernet into subnets with bridges between them which transmitted only those packets that were addressed to another subnet. The bridges operated at the datalink protocol layer so that higher-level protocols such as DECnet and TCP/IP could use the same physical network. This worked well if most traffic was within a subnet, but the Andrew project chose a system architecture that relied on very high performance connections between workstations anywhere on campus and a large central file system, the Andrew File System.

IBM created Token Ring as an alternative to Ethernet. Technically it had several advantages, including better network management and being designed from the start to run over a shielded twisted pair, rather than the coaxial cable in the original Ethernet specification. With the benefit of hindsight we now know that Ethernet was able to overcome its limitations and soon became the dominant standard for campus networks, but at the time IBM Token Ring looked very promising. Since IBM money was paying for much of the Andrew project, the network needed to support Token Ring.

The Andrew Network therefore had to support both Ethernet and Token Ring at the datalink level. Although TCP/IP was the standard, it was also necessary to support the higher-level DECnet protocols over Ethernet. For example, scientific users needed DECnet to connect their VAX computers to the Pittsburgh Supercomputing Center.

Building the network

The network was built by a team led by John Leong. Major contributions came from Computer Science, Electrical and Computer Engineering, and the Pittsburgh Supercomputing Center. Don Smith led the IBM group.



Wiring design used for the Carnegie Mellon network.

The Carnegie Mellon campus network

This greatly oversimplified diagram shows how the campus was wired. Copper wiring was used within the buildings and fiber optics between them.

Diagram by William Arms

Wiring the campus was seen as a long-term investment with the flexibility to adapt to different networking technologies. As the diagram shows, the wiring has a star-shaped topology. The IBM Cabling System was used within buildings, providing high-quality shielded pairs of copper wire. They could be used for either Token Ring or Ethernet protocols. The cables also included telephone circuits. IBM donated the equipment and contributed half a million dollars towards the costs of installation. The wires in each building came together in a wiring closet where there were network interfaces for Token Ring and Ethernet. This provided a Token Ring subnet and an Ethernet subnet in each building. AppleTalk connections were added later. By 1990, there were 1,900 Ethernet, 1,500 Token Ring, and 1,600 AppleTalk ports active.

Leong often described the network as having "an inverted backbone". One spur of each subnet was to the computer center, where they were all connected to a short backbone Ethernet. Routers were used to direct the TCP/IP traffic between the subnets. They were originally developed by the Computer Science and Electrical and Computer Engineering Departments and then the software was ported to IBM PC/ATs. Compared with the nodes on the Dartmouth network, these routers had to handle much higher data rates, but they did not act as terminal concentrators. Separate servers were used as gateways to higher-level protocols, such as AppleTalk.

In practice, there was one major departure from this architecture. Ethernets that carried DECnet traffic were connected to the backbone via bridges, not routers. At the cost of extra load on the backbone, the bridges connected these Ethernets with no constraints on the higher-level protocols.

Connections

To make full use of the network it was desirable that every computer at Carnegie Mellon use the Internet protocols. Support for Unix already existed. The version of Unix that was widely used on campus was the Berkeley Software Distribution (BSD), which included an open source implementation of TCP/IP. BSD was developed on Digital VAX computers. It was standard on Sun workstations and a variant was used by IBM on their Unix workstations.

For the VAX computers, a member of the Computer Science department ported the Internet protocols to run on the VMS operating system. It was distributed for a nominal fee and widely used around the world. This was an early example of the benefits of an open source distribution. Numerous users reported bug fixes that were incorporated into subsequent releases.

The first port of TCP/IP to the IBM PC came from MIT. When we came to deploy it at Carnegie Mellon we encountered a problem. With the Internet protocols, each IP address was allocated to a specific computer. When many copies of the communications software were distributed on floppy disk, we needed a dynamic way to assign addresses. The solution was to extend BootP, a protocol that enabled computers to obtain an IP address from a server. The modern version of BootP is known as DHCP.

In this way, TCP/IP was available for all the most common types of computers at Carnegie Mellon except the Macintosh. The development of TCP/IP for Macintoshes over twisted-pair wiring was a splendid example of cooperation among universities. Carnegie Mellon was one of about eight contributors and at least two start-up companies sold products that came out of this work. TCP/IP service for Macintoshes over the Andrew network was released at the beginning of 1987.

The initial connection of the Andrew network to the external Internet was through the Pittsburgh Supercomputing Center. Since the center was one of the hubs on the NSFnet, the campus network was immediately connected to the national network. Later, a direct connection was made that did not go through the supercomputing center.

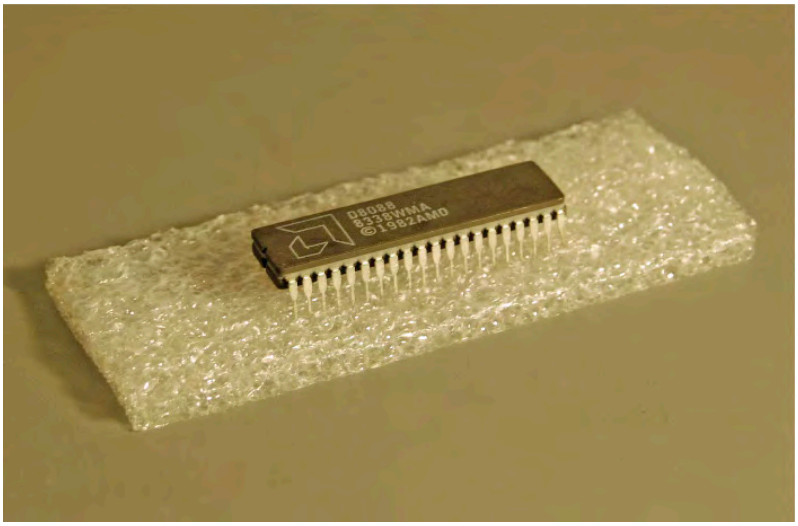
Later developments

In the past thirty years, the network has been continually upgraded. The major developments have been higher speed and greatly increased capacity, beginning with the replacement of the routers and the inverted backbone by high-performance Cisco routers. As the Internet and the TCP/IP family of protocols became widely accepted the need to support alternative protocols diminished. Digital and DECnet no longer exist. Apple uses Ethernet and TCP/IP. Token Ring never succeeded in the market place and was replaced by higher-speed Ethernets. Wired networks are steadily being replaced by wireless networks.

Most of John Leong's team followed the gold rush to Silicon Valley. With their practical experience in engineering a state-of-the-art network, several of them became successful entrepreneurs.

Chapter 5

Networks of Personal Computers



An early microprocessor

Personal computers were made possible by moderately priced microprocessors and memory. This is an Intel 8088 processor, which was used for the IBM Personal Computer in 1981. It has a 16-bit processor but its performance was slowed by an 8-bit external bus.

Photograph by William Arms

Personal Computers and Workstations

Early personal computers

Minicomputers were followed by microcomputers, which soon became known as personal computers. They were made by a new group of companies, such as Tandy, Commodore, Apple, and Acorn, which build the very popular BBC Micro.

The first personal computer that I owned was a Terak. It was a graphics workstation with an elegant Pascal interpreter. The Terak had a 16-bit processor based on Digital's LSI/11 chip set, but most of the early computers used lower cost 8-bit microprocessors. A typical computer of this generation had 4K to 32K of memory, a keyboard, a simple monitor, and two 5-inch floppy disk drives for backup storage. The early monitors were black and white only, but low resolution color monitors followed rapidly. Dot matrix printers from companies such as Paper Tiger and Epson completed the configuration. Tandy's TSR-80 was an early success with hobbyists, but the Apple II was the machine that captured the imagination of schools and universities.

The standard programming language was a simple version of Basic. They were easy machines to program but their success came from the rapid development of a market for software applications. This included general purpose applications, such as word processing, databases, etc. An enormous number of simple programs became available for every level of education. Computer games such as Pacman were a great success, but the application that made the commercial marketplace accept personal computers was the first ever spreadsheet, Visicalc for the Apple II.

Most of the established computer companies missed out on the personal computer and many of them went out of business. IBM was late but when they entered the market they did so with a bang. The original IBM Personal Computer (PC) was only an incremental advance technically, but IBM put its full marketing power behind it. Organizations that had been reluctant to buy personal computers from upstart companies were comfortable buying from IBM. Software developers made the safe choice of developing their products for IBM.

An Apple II+ computer

Most Apple IIs had two floppy disk drives but no hard disk. Floppy disks could be used to distribute software and a software market developed. The computer is running a popular flight simulator.

Photograph by John Miranda

The IBM salesmen had been trained to throw scorn on personal computers and had a hard struggle adapting. At the first Dartmouth presentation, one of our staff took pity on them and fielded questions from the floor while the salesmen took notes. In contrast, the Apple representative was always welcome because she was continually showing us neat things that she did on her own computer.

An IBM personal computer

This is a typical configuration of an early IBM Personal Computer. It has two floppy drives and a dot-matrix printer. The addition of a hard disk with the later model XT transformed the usability and performance. Notice the very basic user interface.

German Federal Archives

Most of the early makers of personal computers could not compete with this onslaught, but a new group of manufacturers emerged selling IBM-compatible computers, popularly known as "clones." In their hurry to bring out the PC, IBM used components that they bought from other companies, such as the Intel 8088 processor and a primitive operating system from Microsoft, a small company that was previously known for its Basic interpreter. I think that Compaq was the first company to build an exact IBM clone. Any software that would run on an IBM PC would run on a Compaq. For example, the dominant spreadsheet for many years was Lotus 1-2-3. The very first release stated on the box that it would run on IBM PCs and Compaqs. Although IBM dominated the personal computer market for many years, they never controlled it. When, in 1987, they introduced an incompatible replacement, the PS/2, it was too late. Microsoft, Intel, and the clones controlled the market.

Lotus 1-2-3 is often credited as being the killer application that made the success of the IBM PC. I think that is an overstatement, but it was very important in forcing the clones to be completely compatible. Previously, every manufacturer had its own operating environment, and software vendors were forced to create separate versions for each of them. I recall visiting Informix, who had a relational database system that ran on all types of computers. In the warehouse they had dozens if not hundreds of different versions packed up ready to ship. It must have been a nightmare. When Bill Gates announced that Microsoft would release a product only if they expected to sell at least 100,000 identical binary copies, the standardization was complete.

An Apple Lisa

This is a Lisa II. It has a built-in hard disk and a single floppy. It was a commercial failure, and deserved to be, but compare its modern windowing interface with the text-based interface on the IBM PC.

Photograph courtesy of Apple Computer

The user interface for these early personal computers were much worse than the good timesharing systems. Some applications, such as Lotus 1-2-3, were well designed but they were controlled by a crude command language. The first personal computer with a modern interface was the Apple Lisa, introduced in 1983. The Lisa was over-priced, unreliable, and woefully slow, but it had a few superb applications, such as the LisaDraw graphics program. Four models were built and I had them all.

Workstations

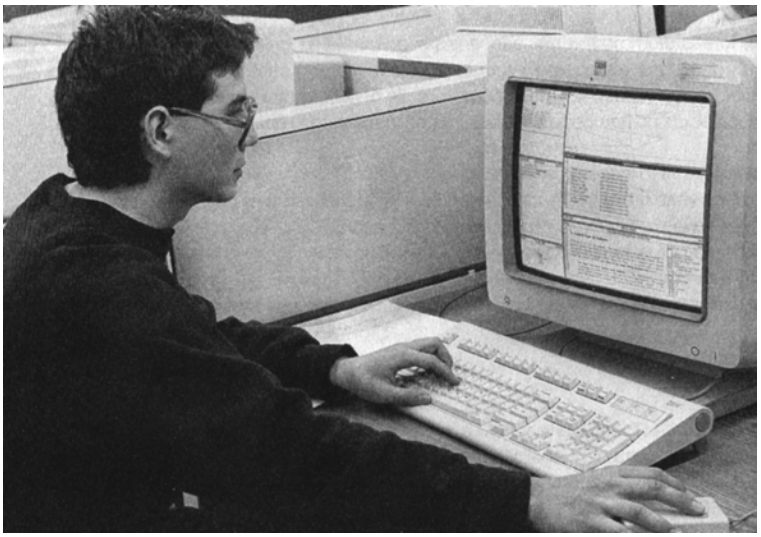
In the mid-1980s there was a rush to build large personal workstations. They were jokingly called "3M" machines, because they had a megapixel display, a megabyte of memory, and a processor that executed one million instructions per second. The early models also had a 10 or 20 Mbyte hard disk and cost about $10,000, which is a million pennies.

The first sales were to engineers and scientists, but prices fell rapidly and sales grew fast. New companies emerged such as Apollo, Silicon Graphics, and Three Rivers Corporation, but they often floundered because of overly ambitious software plans. For instance, the Three Rivers Perq had a beautiful bit-mapped interface but no applications.

Digital had a fast microprocessor, the microVAX, but the company hesitated. Their profits came from minicomputers and they were reluctant to undermine sales of the VAX minicomputers with much cheaper workstations. Unix was developed on Digital computers, but VAX/VMS was Digital's pride and joy, and their support for Unix was always lukewarm. Their later workstations were well engineered, fast, and reasonably priced, but they misjudged the market. Digital, which had been the second largest computer company in the world, never recovered and went out of business a few years later.

The company that came to dominate the workstation market was Sun Microsystems. One of the founders, Bill Joy, was also a principal developer of BSD Unix. Since BSD included the definitive implementation of the Internet protocols, the Sun workstations ran well on the emerging networks. In the early days Sun's business model was to be first to market with advanced hardware and innovative software, often at the expense of reliability. When I joined Carnegie Mellon in 1985 my office computer was a Sun 2. It crashed frequently. This was a mystery until a colleague pointed out that it failed when the sun fell on it in the early afternoon. Later Sun workstations were much more reliable. They were powered by Sun's own SPARC processors.

The Andrew project at Carnegie Mellon, funded by IBM, was built on BSD. Every program that ran on Andrew had to run on standard BSD. When the project began, IBM was not yet ready to ship its own Unix workstation. To everybody's surprise, IBM allowed Carnegie Mellon to buy more than 100 Sun computers for the project.



A Carnegie Mellon student at an IBM RS 6000 workstation in 1986

This computer is running the Andrew interface. The display in the photograph was an experimental display and not available commercially.

Photograph by Kenneth Andreyo

IBM made a marketing mistake in delaying the announcement of their Unix workstation until their engineers were comfortable with the reliability of the hardware and had reworked parts of Unix. As a result they were late to market and the operating system was not quite compatible with BSD. A further mistake was not to offer a high-resolution display with the first release. While IBM was resolving these problems, Sun was consolidating its position as the market leader.

NeXT

Steve Jobs's NeXT computer was a glorious failure. When he was forced out of Apple in 1985 he set out to build a computer with the power of a workstation but the usability of a Macintosh. His target market was education and the business model did not depend on the economies of scale of mass manufacturing.

A NeXT Computer

This photograph is from http://www.johnmirandaphoto.com/next.htm, which has an excellent description of the NeXT computer.

Photograph by John Miranda

Many of the technical concepts were influenced by the several visits that he paid to Carnegie Mellon in 1985 and 1986. When my family and I moved to Pittsburgh we lived for a few weeks in a large house that belonged to Carnegie Mellon. For some of that time, Jobs was also living there. Later I was the first member of the NeXT advisory board, but his main contacts were with members of the Computer Science Department. By them he was persuaded to use Carnegie Mellon's new Unix kernel (known as Mach), and to include a digital signal processor, which made the NeXT computer particularly suitable for applications such as speech recognition.

Technically the NeXT computer was a powerful Unix workstation with a large bit-mapped screen, but it was full of features that were characteristic of Jobs's creativity. It was packaged as an elegant black cube, with few cables visible. When I first visited NeXT in Palo Alto, much of the time was spent in showing off the audio quality and color animations, which were far beyond other machines of the time. Unfortunately, the NeXT machine suffered from another of Jobs's characteristics: it was very late. He would tell you, "It's worth waiting for", but the market did not agree.

Many features of the NeXT computer are now standard but its importance lies in the software. Jobs eventually returned to Apple and the software came with him. Today's Mac OS X is a direct successor to the NeXT operating system. The Interface Builder used to program iPhones and iPads is a descendant of tools built at NeXT to overcome the difficulties that developers had faced in writing programs for the early Macintosh.

The Macintosh at Dartmouth

The dilemma

In the early 1980s, Dartmouth faced a dilemma in deciding how to use personal computers. The success of the timesharing system could have been a barrier to change. It was much more user-friendly than the primitive environments provided by the Apple II or IBM PC; it was much faster for scientific computation; and it was cheaper per user than personal computers. The Kiewit Network already provided a campus email service and access to the library catalog. Yet when we looked at the trends, we saw such rapid improvements in personal computers that they were clearly the path of the future. The Apple II already had a large pool of educational applications and the IBM PC had applications such as word processing and spreadsheets that were far superior to anything that we could produce locally.

Dartmouth had a number of advantages. Dartmouth is large enough to have substantial resources but small enough to be more nimble than larger universities. We had a strong technical staff with close relationships to the faculty. The emphasis on timesharing had made us experts in campus networking. Most importantly, Dartmouth was accustomed to being a leader in academic computing and wanted to remain so.

The solution to this dilemma was to envisage a dual role for a personal computer. A small computer could be used by itself for the tasks that it did well, such as spreadsheets and word processing, but it would also be a terminal to the timesharing system, library catalog, and other network services. Recollections are notoriously unreliable, and I have no records of the process by which this concept emerged, but I remember two key events.

We were all concerned that people at Dartmouth would reject personal computers because of their crude user interfaces, but in late 1982, I heard a leak about a secret Apple project called the Macintosh, and learned extensive technical information about it. When our first Apple sales representative visited us she was horrified to hear of the leak, but after I suggested that we might buy a thousand she arranged for me to visit the Macintosh group in California. I came home convinced that the Macintosh had the potential to succeed at Dartmouth and that Apple was a good company to do business with.

The next event was in spring 1983. The provost, Agnar Pytte, attended a meeting of provosts from other universities where computing strategies were discussed. I provided him with a set of planning slides. One morning soon afterwards he called me to say that he was meeting with the president that afternoon and planned to propose universal ownership of personal computers for Dartmouth freshmen. I hastily put together a short briefing paper. I do not have a copy of that paper, but the main points were to use a personal computer as a terminal to the timeshared computers and the library catalog, to extend the network into every dormitory room, and to use our existing organization to support the initiative. I included an outline budget, which he wisely did not show to the president, but otherwise this paper became the basis for our personal computing plan. Pytte's idea was to begin with the class that entered in fall 1985, but because of a miscommunication when we talked over the telephone I gave him a plan for 1984. He noticed the mistake just before his meeting, gave me a quick call, and kept the 1984 date when he presented the plan to the president.

This paper did not specify which type of computer we would select. I was personally enthusiastic about the Macintosh and lukewarm about the alternatives, but the president came from a commercial background and his ideas about computing were fixated on IBM. IBM wanted us to adopt a stripped computer known as the PC Junior, and Digital made a strong effort to persuade us to adopt their Rainbow computer. The Macintosh project was top secret and I was the only person at Dartmouth who had seen one, but eventually Apple was persuaded to demonstrate the prototype to a group of a dozen people. At the end of the demonstration, Pytte said, "I think that the humanities faculty would like that," and the decision was made. It was typical of Dartmouth that the user interface was the decisive factor in the decision.

An early Macintosh

This picture shows the distinctive shape of the first Macintoshes. It had 128 Kbytes of memory and a single 3½" diskette with a capacity of 400 Kbytes. The small size helped when using the computers in dormitory rooms.

It is hard to recall just how primitive the Macintosh was when it was released in winter 1984. It was slow, with very limited memory, and had only a single diskette, so that copying files was nearly impossible. There were only two applications: MacWrite, a word processor that crashed when it ran out of memory, and a pixel editor called MacPaint. A decent spreadsheet, Microsoft Multiplan, followed soon afterwards, but the terminal emulator, MacTerminal, was not released until late summer.

Yet the Macintosh had some features that appealed enormously to a segment of the academic community. First, it appealed to the eye. The machine itself was cute, the screen display was attractive, and accessories such as the traveling case were well designed and well made. For the first time we had a computer that was intuitive to use. When he first showed me the machine, Dan'l Lewin of Apple dumped the prototype on the table in its traveling case and went to fetch coffee. By the time that he returned I had set up the machine and was moving windows about on the desktop. I still have the coffee mug.

From the day that it was launched, the Macintosh was a joy to use. It was a complete transformation from the command line interfaces used by other computers. Microsoft took ten years to produce a similar interface with Windows 95. Text is central to academic life and the Macintosh was the first moderately priced computer to support full character sets, including diacritics, and to use bitmapped fonts for both the screen and the printer.

Without acceptance by a few universities the Macintosh might well have died as so many other personal computers did during that period. When commercial companies saw the early Macintosh they saw a toy with very little software that was a brute to program. Their price was more than twice the deeply discounted price that we paid. Companies looked at the Macintosh and did what they had always done: they bought from IBM. As the saying went, "Nobody ever lost their job by buying from IBM."

Dan'l Lewin, the head of university marketing, was the person who saved the Macintosh. When I first met him he showed me the draft of an agreement for what became the Apple University Consortium. The basic idea was to offer Macintoshes to selected universities for a very low price, $1,000. At that time, we were wrestling with contracts from IBM's lawyers who wanted to use mainframe concepts for the sale of personal computers. It was a joy to read Lewin's straightforward draft and be invited to suggest changes that would make it more acceptable to universities. Unlike IBM, Apple made no attempt to state what our community would do with the machines or to guarantee how many we would sell, both of which are impossible commitments for universities.

About a dozen leading universities joined the Apple University Consortium before the formal release of the Macintosh, and Apple supported us well over the years. Dartmouth was unusual in that most of the other universities offered Macintoshes as one of several brands of computer that they supported, but we were determined to select one brand and to put all our effort into it. Drexel was the first university to select the Macintosh for all its students, but Apple particularly valued Dartmouth's reputation in educational computing. While IBM and Digital were prepared to make major gifts to gain our business, Apple was more restrained except for the deep discounting. However, they gave us a grant of several machines, which we used as seed machines for faculty to use in education.

The Macintosh at Dartmouth

When Dartmouth decided to adopt the Macintosh, the provost presented the plan to every single committee of which he was a member, but that does not mean that everybody welcomed the initiative. He and I had to endure a fairly rough session at a meeting of the university faculty, and the president never really accepted that we had rejected his friends at IBM.

The first year, 1984, was hectic. The resources of the computing center were focused on the arrival of the freshmen in September, but all the usual services had to be kept running. Converting the network to AppleTalk was a major engineering feat. At the same time, the telecommunications team had to wire the dormitories, install new nodes, and make the cables to connect Macintoshes to the wall outlets. The communications package of MacTerminal and the special cable were slightly late, but were delivered to the students a few weeks after the beginning of the term. Logistics were a challenge. The computing center had a small store selling manuals and supplies. This was transformed into a computer store supplying Macintoshes and IBM PCs, with their software and supplies. Technicians had to be trained in repair work. The handout of the computers to the freshmen was turned into an event, with a convoy of trucks delivering them from the warehouse.

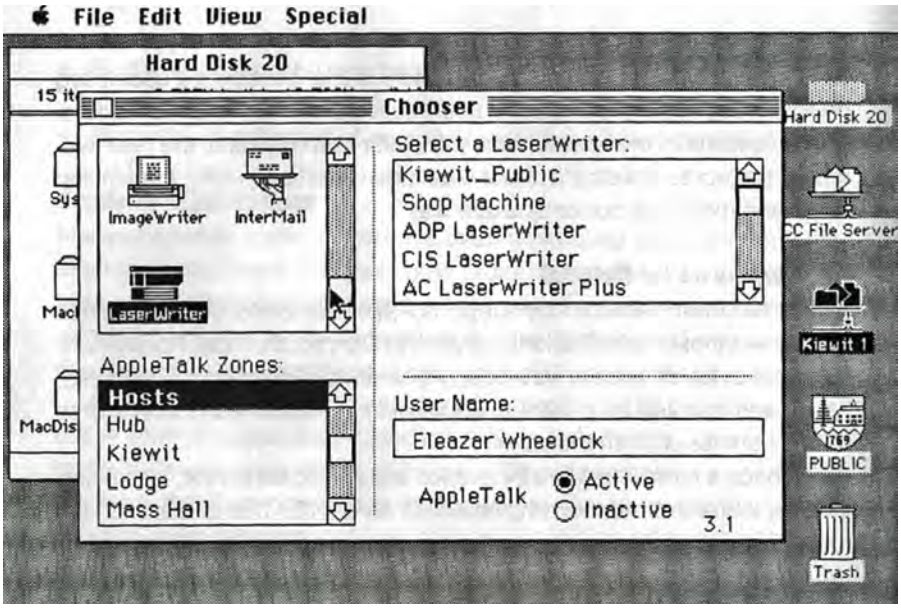
For those of us who were advocates of the Macintosh, the challenge was to balance the long-term potential against the deficiencies, which we hoped were short term. The enthusiasm of a few core faculty helped solve the shortage of good applications. Although the early Macintosh was an awkward machine to program, the qual ity of the QuickDraw tools enabled determined individuals to create good programs quite quickly. We were able to support them with the machines that we had received from Apple and grants from several private foundations

Dartmouth freshmen collect their Macintoshes

In 1984, the standard package was a Macintosh with MacWrite, and MacPaint; a communications package was delivered later. Many students also chose a carrying case and a printer. Students were urged to buy computers, and between 75 percent and 80 percent of the freshmen did so. Owning a computer was made a requirement in 1991, but it did not have to be a Macintosh.

Photograph by Stuart Bratesman, © 1984 Stuart Bratesman - All rights reserved

Apple worked effectively to overcome many of the limitations without compromising the overall design. At the first meeting of the Apple University Consortium, Jobs spoke of several developments, notably the plans for a small laser printer, which was the first for any personal computer. Extending the memory to four times its original size and adding a hard disk solved two crucial problems.



DarTerminal

This is a DarTerminal screen dump. It shows how Apple's Chooser could be used to connect to computers and services on the Kiewit Network.

Screen image by Rich Brown

Over the next few years Dartmouth used its technical resources to provide networked services that augmented those provided by Apple. The original MacTerminal was replaced by a more flexible program, DarTerminal, and the Avatar distributed editor was ported to the Macintosh. Dartmouth's Fetch program, an FTP client, is still going strong as a file transfer program for Macintoshes. A new mail system, BlitzMail, was so well received that there was dismay when it was withdrawn in 2011. A group of universities collaborated in providing AppleTalk service over dial-up telephone lines. Many of these developments took place after I moved to Carnegie Mellon in summer 1985.

The decision to select one type of computer and support it well proved a great success, but Dartmouth was never exclusively a Macintosh campus. IBM PCs were always preferred by the business school. Other individuals chose them or later the Windows clones. From the earliest days, the computer store sold IBM PCs and the computing center steadily extended the distributed environment to support them. Networked ports in the dormitories could be configured for asynchronous connections and later for Ethernet. BlitzMail and the Avatar were ported to the IBM PC. A report in 1991 estimated that there were about 10,000 computers on campus of which 90 percent were Macintoshes.

Carnegie Mellon and the Andrew Project

Planning

At Carnegie Mellon, the initiative came from the top. The president, Richard Cyert, summarized the university's strategy as "a campus saturated with computing." He saw that state-of-the-art computing could benefit every aspect of the university, including all academic departments, the library, and the administration. He saw research opportunities in every field and the potential to transform education. As a canny businessman he rightly anticipated that an aggressive computing strategy would receive financial support from corporations, government agencies, and private foundations. In his keynote address to the 1986 EDUCOM Conference, Herbert Simon aptly described the strategy as "computing by immersion." Provide faculty and students with a great array of computing and they will use it creatively.

The timing of the plan was based on a deep understanding of trends in computing. In 1979, Allen Newell published a report known as the Spice Report. This report extrapolated hardware trends to predict the spectacular advances in large personal computers that took place during the 1980s. It would soon be possible to build a computer on a single chip that would be as powerful as the super-minicomputers. These large personal computers were often called "workstations".

The Preliminary Report on the Future of Computing at Carnegie Mellon University

Photograph by William Arms

Newell later chaired the Task Force for the Future of Computing, which articulated how Carnegie Mellon could use these advances. The Xerox Palo Alto Research Center had already demonstrated how workstations could be linked by a high-speed network to form a new model of computing, but this was used by only a small

number of well-funded computer scientists. The vision was to recognize that this style of computing was appropriate for an entire university and that declining hardware costs would make it affordable. Carnegie Mellon set out to build such a system.

To achieve the ambition, Carnegie Mellon needed an industrial sponsor and they found it in Lewis Branscomb, the chief scientist of IBM. With his support IBM gave the university a five-year grant, later renewed, to build an advanced campus computing environment known as Andrew. The grant was huge. It included an R&D center, the Information Technology Center (ITC) with thirty computer scientists and engineers, very large donations of equipment, support for educational computing initiatives, and social science studies of the impact. A separate grant helped build the campus network.

People

In writing about the Andrew project it is important to stress that this was a university-wide initiative. The various groups that contributed are too numerous to list, but two were so important that their leaders had university titles. Howard Wactlar, Vice Provost for Research Computing, was responsible for the extensive computing facilities in the Computer Science Department. He had exceptional insight into the boundary between research and practice, and how to create partnerships with industry. Michael Levine, a physicist, was one of the founding directors of the Pittsburgh Supercomputing Center. He had the title Associate Provost for Scientific Computing and was a vigorous advocate for those researchers who needed computing facilities beyond those that a Unix-based project could provide.

My predecessor, Doug Van Houweling, was a key person in creating the strong relationship with IBM and recruited Jim Morris to be the first director of the ITC. They were ably supported by John Howard, the senior IBMer on campus. While the ITC created the prototype system and was a catalyst for the rest of the university, components were contributed by many departments, by other universities, and by several manufacturers.

In conjunction with the Andrew project, Carnegie Mellon created the Center for the Design of Educational Computing, under the leadership first of Jill Larkin and later Preston Covey. Year after year, the center received at least one of the annual EDUCOM awards for excellence in educational software.

As the project moved past the prototype stage, responsibility moved to the university computing center. Two people transferred from the ITC to lead this effort: Bob Cosgrove who had responsibility for the servers and the centrally run systems, and John Leong who built the campus network. Without their efforts we would never have overcome the myriad of technical challenges that lie between a research project and a stable production environment. Later, when Cosgrove left and the IBM funding expired, Leong became the overall technical director.

The Andrew project

In 1982 Carnegie Mellon was already a leader in computer science and campus computing. At the time of the Newell report, 7,800 faculty, students, and staff had 1,090 terminals connected to a variety of timeshared computers. The computing center ran six large DEC-20 computers and Computer Science had several more. Numerous departmental centers ran VAXes. Several departments and the computing center operated a large Ethernet using DECnet protocols. The business school became the domain of the IBM PC. The nucleus of a modern workstation environment existed in Computer Science, where 18 Xerox Altos, 44 Perqs, and a Dover laser printer were connected over an experimental 3 Mbit/sec Ethernet. The challenge was to transform this nucleus, so that it would have a major impact on research, on the administration of the university, and especially on students.

When the ITC began work, none of the components that are now considered fundamental to distributed computing existed, except in research laboratories, yet the ITC's aim was to support thousands of personal computers. Ethernet was still experimental; workstations such as the Sun 1 were expensive, slow, and unreliable; Unix was considered impossible to support without an army of system programmers; and the Apple Lisa and Macintosh computers had not yet shown the advantages of a graphical user interface. Nobody had experience with large-scale distributed computing.

The members of the ITC had to work in all these fields and more. They developed a campus-wide file system, a window manager and applications toolkit, and did much of the work in creating the campus network. Morris and his colleagues also recognized that building the technical framework was only part of their task. They had to help people to use it. For this purpose, they supported a number of faculty initiatives, and wrote an excellent mail and messaging system.

The ITC was funded by IBM, but its work was not restricted to IBM equipment and software. The Andrew software was based on BSD Unix and ran on IBM, Sun, and Digital workstations. The port to the Sun 3 had an interesting bug. The Sun 3 was generally very reliable, but it crashed when running the Andrew window manager. A member of the ITC traced the problem to a memory boundary problem with a specific sequence of instructions that Sun's software never used. He reverse engineered the paging firmware, made a minor hardware modification, and fixed the problem. Unfortunately, while doing this he was being paid from the IBM grant; IBM was not amused.

Network services

By themselves, personal computers could not replace timesharing. Timeshared computers shared much more than the central processor. Their central file systems allowed users to store programs and data, to share printers and other peripherals, and to communicate with each other. One of the tasks of the Andrew project was to provide this sharing through network services. The Andrew File System consisted of a number of server computers, each with several large disks. A Unix workstation on the network saw one large file system in which there was no distinction amongst the servers, nor between files stored locally on the workstation and files stored

The Andrew File System

The photograph shows the Andrew File System in 1987. The servers are Sun 2s. Each server has three Fujitsu Eagle disk drives, each with a capacity of more than 400 Mbytes.

Photograph by William Arms

on the file system. In fall 1989, the main campus file system had 15 server computers and more than 30 gigabytes of storage. It was used by 5,000 users every month. In addition to the central file system, which was

available to the entire campus, there were several operated by departments.

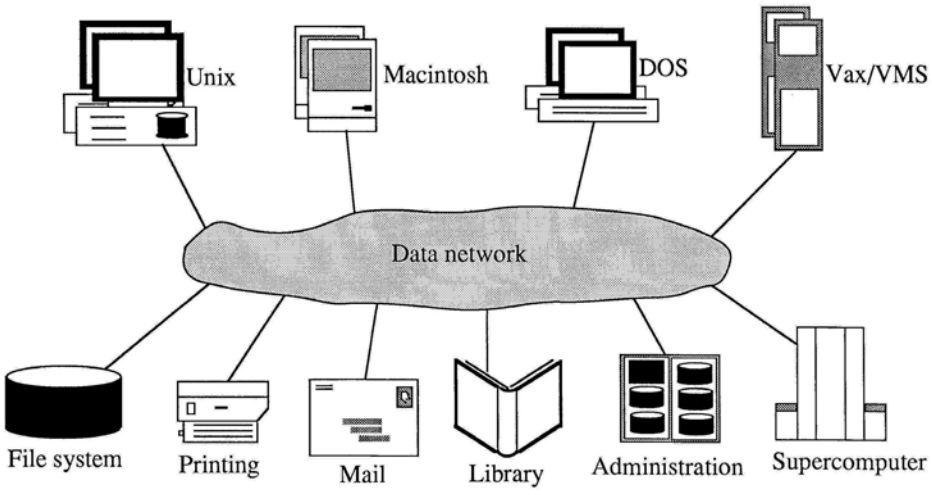
The Andrew message system supported electronic mail and bulletin boards. It was created by the ITC and extended by the computing center. The mail delivery used small post office computers and the user interface ran on personal computers. Mail between users of the Andrew File System was deposited directly into personal mailboxes. Mail to other mail services, including the Internet and Bitnet, was handled by the post office computers. The software had good algorithms for resolving ambiguous or duplicate names, and for handling difficulties. There were separate user interfaces for different types of personal computer. The interface for Unix workstations supported structured text, bit-mapped images, and other formats. Other interfaces were tailored for IBM PCs, Macintoshes, and simple ASCII terminals. Carnegie Mellon was an early advocate of the IMAP mail protocol.

The applications gap

These were tremendous achievements, but there was a gap. It was the same gap that eventually doomed time-sharing at Dartmouth. Unix was undoubtedly the right choice for the system's infrastructure that the Andrew project built, but Unix has always been weak on applications.

Many academic units were opposed to the emphasis on Unix workstations and many who supported the overall vision held back from using the new software until it had proved itself. People refused to commit to Andrew-Unix because the best applications in their areas ran on other computers. Macintoshes and IBM PCs were everywhere; the School of Industrial Administration, and the School of Urban and Public Affairs selected PCs; the new School of Computer Science and the department of Electrical and Computer Engineering used Unix but not the Andrew window manger and tool kit; humanities and social sciences chose Macintoshes; engineering and science departments had a combination of Unix and VMS workstations.

My major contribution to the Andrew project was to recognize this gap and refocus the resources of the computing center to tackle it. To their credit, the IBMers on campus also recognized the situation and were supportive. If today you search the web for Andrew, you will find a narrow description of the Unix software that the ITC developed, but a 1986 brochure from IBM's Academic Information Systems is much broader. Naturally it emphasizes what IBM contributed, but its focus is on networking, and the importance of interoperating with many types of computers and with other universities. In the brochure, I am quoted as saying, "...the plans assumed that there would be nothing else on campus except Andrew. [The concept] was comparable to building an expressway across Wyoming, working with virgin territory, when in fact we were talking about running one through something like Chicago. The unavoidable interrelationship with other computer systems on campus was not foreseen."



Andrew Plus

This is a diagram that I used to describe Carnegie Mellon computing in the late 1980s. The top row shows the supported types of personal computers and workstations. The bottom row shows some of the services that these personal computers could connect to over the network. It does not show the departmental computing centers. Unix computers could connect directly to the networked services. IBM PCs and Macintoshes connected via proxy servers.

Diagram by William Arms

When personal computers were first developed, most people expected that they would supplement timesharing, not replace it. A common question was "what balance would emerge?", but when the central DEC-20 computers were withdrawn in 1988, they were not replaced.

Although personal computers and networked servers dominated the environment, several large computers remained. The largest was the Cray computer at the Pittsburgh Supercomputing Center, operated jointly with the University of Pittsburgh. The library, under the leadership of Thomas Michalak, was a leader in building online information systems. Taking advantage of a lightly used IBM mainframe, the library provided a single user interface to the library's catalog, secondary information services, reference materials, and university information such as the staff directory. Most administrative data processing continued to be done on VAX computers.

Deployment

To build this environment we had to create a new type of organization. Originally, the Andrew initiative had by-passed the computing center and the staff were naturally dispirited. One of my first actions was to reverse the decision to build a separate team to deploy the new environment and to give the responsibility to the computing center.

The transition from timesharing to personal computing was an enormous wrench. The mature and well-loved DEC-20 systems received little attention while the new environment was developed. In 1988, when they were withdrawn, the distributed environment was far from complete. Many components were rough and ready, some were incomplete or temporary, performance was erratic, and many skilled people were needed to carry out tasks that later were routine. In a 1989 survey, students ranked computing well ahead of all other student services at the university, yet several faculty reports were disappointed at the impact on education. Carnegie Mellon was probably the most computer-intensive university in the world, yet shortage of resources was a constant complaint.

The planning papers rightly forecast the enormous demand for personal computers and gave high priority to ensuring that everybody had access to one. In 1981 there was one terminal on campus for every eight people. In 1990 there were as many computers as faculty, students, and staff combined. Amongst faculty, computer usage was almost universal.

As the Andrew project gathered momentum, many of us wanted to require all students to own a computer. The idea was discussed on several occasions, but the university was never prepared to add the cost of a personal computer to the high cost of tuition. Many students, however, had their own computers and the percentage grew year by year. A survey of freshmen in fall 1989 found that 55 percent either owned a computer or were planning to buy one. In addition, the university operated public computing laboratories with almost 700 personal computers for the students. Half were maintained centrally and half by departments. The 1989 survey found 210 Unix workstations, 269 Macintoshes (including 46 large Mac IIs), and 210 IBM personal computers in public laboroties. For personal ownership, the most popular computer was the Macintosh, and every year the university sold about 1,200 Macintosh computers through the campus store. People turned to the Macintosh and the IBM PC as a straightforward way to get their work done, and were delighted by the stream of innovative software developed by the commercial market.

Reflections

The announcement of the Andrew project generated a wave of publicity in both the press and television. In the hyperbole, Carnegie Mellon forgot to manage expectations. Impossible predictions were made about the transformation of education that would come out of this project and how the technology would sweep the market.

Years later a colleague at Cornell asked me if Andrew was a success. I was too close to the project to give an impartial answer but here is an attempt.

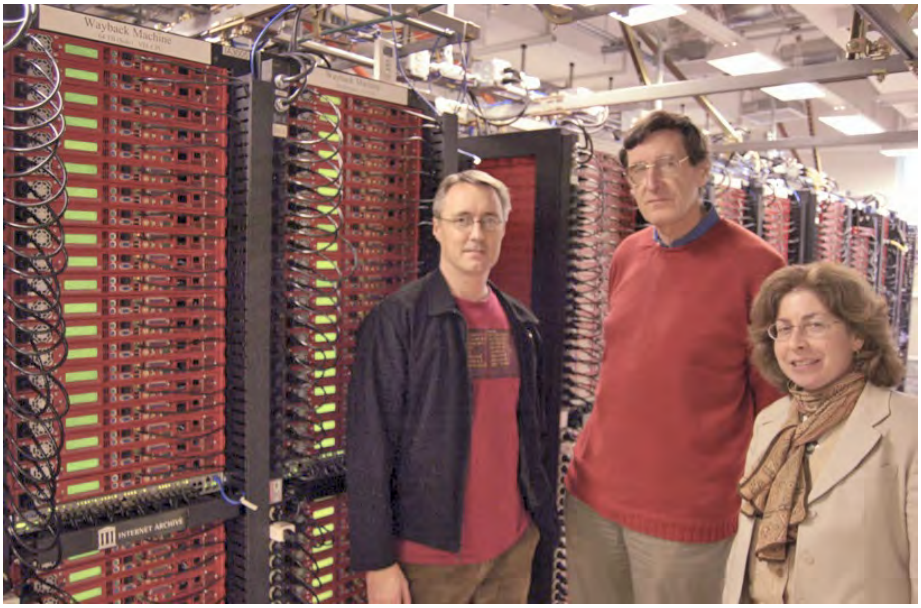
The 1980s were a good decade for Carnegie Mellon by almost every measure: quality of students, research volume, financial strength, and so on. They all improved dramatically and the emphasis on computing was a major contributor. This strength is seen today everywhere on campus.

If the early planning papers are contrasted with what was achieved, the picture which emerges is that the overall strategy advocated in 1982 was remarkably accurate. The forecasts of the tactical steps which would be taken were less accurate. The biggest area not anticipated was the impact of commercial software running on small personal computers. The Andrew Network, the Andrew File System, the message system, and other servers were technical triumphs. For the first time, a major organization ran its computing services without a large central computer. There were also disappointments. The Andrew tool kit and user interface were never widely used, though they had impacts on other projects, and IBM never developed the commercial products that we hoped would emerge from our joint efforts.

At the time we were often swamped by the challenges, both technical and organizational. Now, with the distance of time, it is clear that the strategy of saturating the campus with computing worked brilliantly. Everybody involved can be proud of the outcome.

Chapter 6

Modern Times



A server farm

Personal computers, networked services, the web, cloud computing, and server farms have replaced central computers as the heart of academic computing. This photograph shows a server farm at the Internet Archive in 2009. The Internet Archive is a not-for-profit organization with close ties to academia. Its founder is Brewster Kahle.

Photograph by Felix Weigel

Academic Computing Today

The changing landscape

My role in academic computing changed when I left Carnegie Mellon in 1995. After seventeen years as a computing director I became a user again. With the major exception of digital libraries, I was no longer an insider. This section describes some of the trends that I see as an outsider. Undoubtedly my different viewpoint has obscured important developments, but there seems to have been a fundamental change in academic computing.

The underlying theme of the early years of academic computing was that universities were different. Because the computing industry was not providing the systems that they needed, they built their own. More recently, however, end-user computing has become universal and academic computing has merged back into the commercial mainstream. The projects of the 1980s brought an end to the period in which universities created their own computing environments. The excellent computing that Cornell now provides for its faculty and students is built almost entirely upon standard components. The computer on my lap, the networks it connects to, the programs that I run, the email and web servers are commercial products that are available to everybody.

In 1990, in a review of the Andrew project, I wrote:

> "As we begin the next decade, nothing on the horizon suggests any change as dramatic as the move from timesharing. The Andrew environment, with personal computers and shared information, appears to be the right model for the years ahead. Many skilled people are needed to refine and improve this environment, but assuredly the next few years will see ever improving computing, which will be based around cheaper and more powerful hardware, better system software, more elegant applications, more dependable service, and a higher level of sophistication amongst users. The university will continue to wrestle with the organizational and financial issues, and will eventually find a balance between decentralized control and university-wide coordination.

> "However, the next decade will see much more than an elegant completion to the Andrew project. If we simply extend hardware trends for ten years, we can see that personal computers will execute more than 100 million instructions per second and have screens so good that reading from them will be as pleasant as a printed book, that much of the campus network will be operating at speeds measured in gigabits per second, that it will be cheaper to store almost everything on computer disks than on paper, and that the distinction between video technology and computing will be blurred. These are more than quantitative changes; they are opportunities to open up new areas. The university has vigorous programs in many areas which are poised to take advantage of more powerful computing. These include electronic libraries, speech recognition, natural language processing, chess playing, image processing, and extensive areas of scientific computation. The next decade will see many of these move into the every day life of the campus. Surely, also, the next decade will see developments in new areas that nobody yet foresees."

The final sentence of the quotation has proved abundantly correct. Nobody predicted the uses that would be made of this new world of computing. Less than five years after this sentence was written, the web began its breakneck expansion. Soon afterwards mobile computing began its equally dizzy growth. Both were predicted in general terms, but nobody could imagine their impact. Fifty years ago computing was for highly skilled specialists; twenty-five years ago it was available to the members of well-funded organizations such as universities; today it is universal.

The web

Universities did not create the web, though they were significant contributors. In the early 1990s there were several competing systems for distributing information on the Internet. They included Gopher from the University of Minnesota, the World Wide Web from CERN in Geneva, Z39.50 from the library community, and WAIS by Brewster Kahle, which used a modified version of Z39.50. In retrospect, the greatest strength of the web was its simplicity. Z39.50 failed because it tried to do too much. It assumed that nobody would place valuable information online unless it was protected by an authorization system. The early web was particularly easy to use. I personally created Carnegie Mellon's first home page in about an hour.

An early browser

This is a screen dump from the Netscape browser in 1995. This was the first time that I saw a well-known publication place some of its content on a web site.

Screen image by William Arms

The browser that established the popularity of the web was Mosaic from the University of Illinois, released in 1993. It brought proper fonts, color, and images to the Internet and people loved it. Mosaic was initially developed for Unix workstations but it was rapidly ported to all standard personal computers. The same group at the University of Illinois also developed the web server that is now called Apache.

Most of the web search services, except Altavista, began as university projects or used university search engines, including Infoseek, Lycos, Yahoo (which began as a catalog), and Google, but they all rapidly became start-up companies. More recently, Facebook began at a university but it quickly became a start-up company. MIT was the founder of the World Wide Web Consortium, which has played a major role in standardizing and enhancing the web technology, but overall, universities have been users of the web rather than the creators of the technology.

Open source software

It is hard to exaggerate the importance of open source software in academic computing. Many articles have stressed the benefits of collaboration and the excellence of the best open source software, such as the Linux operating system, Eclipse development environment, Apache web server, Python programming language, Lucene search engine, and the Hadoop-distributed file system and map/reduce engine. The zero cost is of course important, but the availability is even more so.

In my software course at Cornell, I never worry that the students do not have access to the software that they need. Students can download the open source packages onto their own computers and be up and running immediately. Many of the best students never buy any software. They rely on what comes with their computer and what they can download for free. We are educating a generation of students who are experts in the open source packages, but have little knowledge of the commercial alternatives.

Rather surprisingly, universities are not very active in creating open source software. All of the packages listed above are maintained by not-for-profit organizations. Individuals from universities may be contributors, but they get no academic reward for such activities, whereas many corporations contribute staff time as a substitute for building their own software. For instance, Hadoop provides companies that lack Google's expertise with an alternative platform on which to build very large Internet services. Facebook could never have grown so fast without it.

University research and spin-off companies

This is a remarkable time for technology transfer from computing research to the outside world. Ideas developed in universities are making their way into the marketplace very rapidly. Computer science and electrical engineering are in a particularly productive period, where many long-standing research areas have matured into practical products, but people in all disciplines are finding high-powered computing to be a never-ending source of innovation and entrepreneurship.

A dominant theme is the spin-off, a start-up company that individuals create based on their university research. Spin-offs are not new. As early as the 1950s, Stanford University was encouraging the growth of Silicon Valley, while Route 128 around Boston was synonymous with spin-offs from MIT and Harvard. In recent years, the pace has changed as venture capitalists became ever more willing to fund start-ups early in their life cycle. Sun Microsystems, which combined expertise from Stanford and Berkeley, was a good example. Mosaic was developed at the University of Illinois, but a year later, moved to a start-up, Netscape. Search engines such as Lycos (Carnegie Mellon) and Google (Stanford) began as research projects but rapidly formed spin-offs.

Over-exuberance about the potential of the Internet led to a speculative bubble, the dot.com boom, and its stock-market crash in 2000; but the success of companies such as Microsoft, Amazon, Google, and Facebook

are an inspiration for individuals to turn their ideas into products and bring them to market. Each of these four companies continued to be led by its technical founders. In the past there was an attitude that companies should be led by people with business and financial backgrounds, but today's students have realized that founders who come from a technical background can be successful entrepreneurs.

Computing in Education

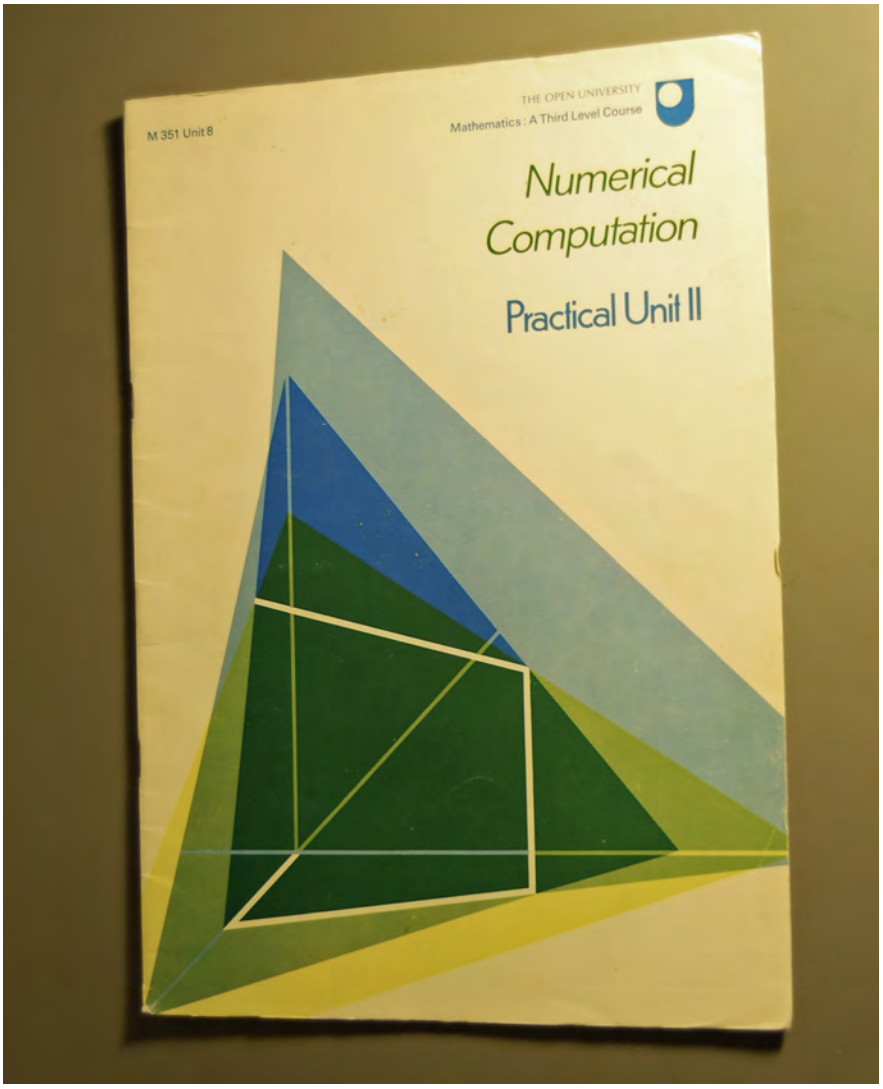
The advocates of every project described in this narrative hoped to have a major impact on education. Dartmouth Basic was designed for students; Athena at MIT and Andrew at Carnegie Mellon were inspired by the goals of better education; when companies such as IBM and Apple gave personal computers to universities they expected to create educational breakthroughs. Meanwhile, every university has its own programs to support innovative courses, and organizations such as the National Science Foundation and the Howard Hughes Medical Institute have well-funded initiatives. During the dot.com boom several universities created web start-up companies. I was a director of eCornell at Cornell.

The talent and energy behind these efforts has been remarkable, but overall the impact has been disappointing. A Carnegie Mellon survey in 1989 found that 44 percent of the faculty used computing in their courses but only 9 percent used programs that had been written for instruction. The real success of computing in higher education comes from faculty offering students the same computing tools that they use for their own research. In addition, we must not ignore the increased productivity provided by simple tools such as word processing, graphics, email, course web sites, and bulletin boards, and the value of online access to the library and other sources of information.

With all the disappointments it is rash to be too optimistic, but there are signs that times are changing. The first sign is financial. Our universities are becoming unaffordable. Partly this is because of inefficiencies, but the underlying cause is more fundamental. Economists call it the cost disease of service industries. An organization that depends on large numbers of well-paid professionals cannot increase its productivity unless it changes its mode of work. As a result it becomes steadily more expensive relative to the overall cost of living. Higher education in the United States is under relentless financial pressure and people are becoming increasingly willing to look for alternatives.

A second sign is that web-based distance education courses are being offered by a wide variety of organizations. Some are of dubious academic merit, but some are excellent. Cornell gives full academic credit for courses offered through the summer school program, and there is increasing willingness to consider other courses that are partially or fully online. We are slowly building an understanding of what works in our culture.

The past few years have seen the introduction of massive open online courses (MOOCs). As usual the hyperbole and enthusiasm have run ahead of the actual achievements, but these courses have some impressive features. Most importantly their advocates are leading faculty from some of our top universities. They include experts in those branches of artificial intelligence that are used to construct courses that need minimal human intervention.



An Open University course unit from about 1975

Photograph by William Arms

My own thinking is strongly influenced by the years that I spent at the British Open University in the 1970s. The Open University does not follow the traditional format of residential education. It was founded to serve adult students, living at home, usually with regular jobs. In the United States this huge group of students is served by a mixed bag of community colleges and for-profit schools, but only too often the result is little education and large debts.

Although the Open University always has had the latest technology available, it is cautious in its use of technology. When I was there, forty years ago, we had a dedicated BBC television studio, but the most effective technology that we had was printed course materials, backed up by human tutors. Nowadays, distance education has few technical barriers. We can assume that our students and faculty have good computers and network connections, and that they are skilled users of them. The technical tools that faculty need to create educational materials are at our fingertips.

Sometime in the future these threads are going to come together and we will see high-quality degree programs based on educational technology. Nobody can tell whether they will be developed by existing universities or by new organizations. The most obvious opportunity is to provide high-quality education for part-time, non-residential students. The developments of academic computing have been led by the elite residential universities, but the greatest benefits may be to people who have never been able to attend those universities.

Sources

In writing this memoir, I began by putting down what I can remember. I then searched for contemporary documents to check my memory of facts and dates. The information that I have been able to find is often incomplete and unbalanced. For example, there is a great deal of information about Multics at MIT, but very little about the Dartmouth Time Sharing System.

Two of the three EDUCOM books cover topics that are covered in this narrative. They are:

Campus computing strategies, edited by John W. McCredie. Bedford MA: Digital Press. 1983

Campus networking strategies, edited by Caroline R. Arms. Bedford MA: Digital Press. 1988

Many of the details about computing at Dartmouth come from the reports and brochures that the Kiewit Computation Center produced intermittently. I have the publications from 1969-71, 1973-76, 1985, and 1986.

IBM's Academic Information Systems division produced a well-balanced brochure on the Andrew project:

Carnegie Mellon University Reaching for World Leadership in Educational Computing and Communications, IBM. 1986

Many of the facts and figures about Carnegie Mellon are drawn from a 1986 brochure and a paper that I wrote when Richard Cyert resigned after nineteen years as president:

Arms, William Y., Reflections on Andrew, EDUCOM Review 25(3):33-43. 1990

Wikipedia has technical articles on many of the computer systems. Some are excellent, e.g., the article on PDP-11 minicomputers, but there are some notable gaps. Wikipedia says very little about how the computers were used and next to nothing on their impact.

Errors and mistakes

Much of this material is my memory of events that happened many years ago. I am sure that there are mistakes. Please send me corrections whether of facts, misunderstandings, or failures to give credit to the right people.

William Y. Arms
Cornell University
wya@cs.cornell.edu

02

KING

The Early Years of Academic Computing:

A Memoir by Kenneth M. King

This is from a collection of reflections/memoirs concerning the early years of academic computing, emphasizing the period from the 1950s to the 1990s when universities developed their own computing environments.

Initial Release: June 2014

Published by The Internet-First University Press
Copy editor and proofreader: Dianne Ferriss
The entire incremental book is openly available at http://hdl.handle.net/1813/36810

Books and Articles Collection – http://ecommons.library.cornell.edu/handle/1813/63
The Internet-First University Press – http://ecommons.library.cornell.edu/handle/1813/62

B. Memoir by Kenneth M. King

Preface

The two most serious oriental curses are reputed to be: "May you live in interesting times and may your spouse understand you". People involved in University Computing in the early years certainly lived in interesting times. I have long believed that Universities have not gotten enough credit for the many things that they did to advance technology and its applications. I believe that Bill Arm's memoir, providing an inside view of the development of time-sharing and distributed computing, two huge advances in the development of operating systems and improving the human interface to computers, is a great start at chronicling some of those contributions. When Bill Arms approached Bob Cooke and me about publishing his memoir in the Internet-First University Press (IFUP), I suggested that he try to create an "incremental book" by persuading other pioneers to contribute their memoirs. The incremental book idea was developed by Bob Cooke as part of our joint effort to get scholarly information on the Internet. The idea was that we would publish chapters or sections in a book on the Internet as they are finished, rather than wait until the whole book is completed. Bill suggested that I develop my own memoir and that we then jointly approach other pioneers to create an incremental book composed of memoirs by people who labored at Universities and may recall events and contributions that helped change the world. This second section in our book covers my observations about life in the trenches at Universities during the period between 1953 and 1993. The book that we hope to create is possibly an effort not unlike the story about an elephant touched by blind men. People that we hope will contribute will certainly have different slants on major events at the Universities that helped change the world.

Contents

Chapter 1

Computing at the Watson Laboratory at Columbia University in the 1950s

My involvement with computing at Universities spanned the years from 1953 to 1993. During that period Universities were responsible for many major advances in computing technology, and University Computer Center staff and University Faculty played a big role in many of those advances. There were four major thrusts in the effort to advance computer technology: efforts to improve hardware performance; efforts to improve computer throughput (primarily directed toward improving operating systems); efforts to provide network access to computers; and efforts to improve the productivity, ease of use, and tools available for people who were programming computers. In the early years, hardware performance and throughput trumped efforts to improve the human interface, if those improvements came at the expense of throughput.

During the 1950s, there were debates on the issue of whether or not Universities should be involved in building their own computer hardware. The Institute for Advanced Study at Princeton built the IAS computer following a design by John Von Neumann from 1945 to 1951, the University of Pennsylvania built the Eniac computer in 1947, and the University of Chicago was building Maniac computers until the early 1960s. By the end of the 50s it became clear that it was impossible to compete with commercial vendors of hardware, and Universities stopped building their own computers. However, people who started a number of commercial hardware companies had deep connections to University Engineering Schools. Over the years that spanned my career, University contributions to the advance of technology were primarily in the areas of computer operating systems, application packages, languages, networking, and the computer human interface.

I became involved with computing as a graduate student in the Physics Department at Columbia University. The IBM Company maintained and funded The Watson Scientific Computing Laboratory at Columbia University, and up until the early 50s the laboratory was stocked with IBM's latest and greatest commercial hardware, primarily tabulating machines. It was at the time one of a few premier University Computer Centers in the world, and IBM provided instruction in programming to its scientific customers at the Laboratory. In 1953 I wanted to solve some interesting problems in Quantum Mechanics, and IBM was building a new computer at the Laboratory for the Navy, called the NORC (Naval Ordinance Research Calculator). From 1954 to 1958 it was the world's fastest computer. It had a whopping 2000 words of cathode ray tube storage with a memory access time of 8 microseconds, an online card reader, card punch, line printer, and some magnetic tape drives. On a good day it could execute about 20,000 arithmetic operations per second. It was programmed in decimal machine language and during its construction I and another graduate student had access to it at odd hours. For its dedication in December 1954 I computed a table of logarithms in a few minutes that circa 1600, Napier had spent many years computing. Unfortunately for Napier, his logarithm tables were necessary to support navigation across the seas, and the age of exploration could not be postponed 400 years waiting for the NORC. I also computed pi and e, the base of the natural logarithm system to a million places, in something like 20 minutes, because John Von Neumann, who was attending the dedication, was interested in testing the digits for randomness. As predicted by theory they were indeed random. In the 50s and 60s, using the most advanced computing technology available at the time, Wallace Eckert and Harry Smith spent years at Columbia University computing the orbit of the moon to enough precision to support the Apollo space program. For computers with small memories backed by tape drives, any program whose data didn't fit into the available memory

proceeded at close to tape access speed. Today this calculation could be done on a laptop in minutes with our gigabyte memories. Unfortunately for them the age of space exploration couldn't wait.

When the NORC left at the end of 1954, the equipment remaining at the Watson Laboratory supported my computing activities, along with a slide rule that every physics student carried at that time. The principle-computing engine in 1954 was an IBM Card Programmed Calculator (CPC), which consisted of an IBM 402 tabulating machine connected to an IBM 604 electronic computer and a card punch. The tabulating machine was controlled by a large removable wired program board, and the 604 was capable of executing 40 program steps on the data on each card fed into it by the 402, and punching the result onto another card. It was altogether a fun machine to work with and capable of executing about 25 arithmetic operations per second. In 1955 the CPC was augmented by a pair of IBM 650s. The 650 was a drum machine containing 2000 words of storage, and each instruction specified an operation to be performed on data at some address on the drum, followed by the address of the next instruction in the program. A drum was a rotating cylinder coated with a magnetic material. The drum rotated at 2500 revolutions per minute. The challenge was to locate the program steps and data on the drum so you didn't have to wait for the drum to rotate all the way around to find the data for the current instruction, or for the next instruction to be executed. Carefully programmed, it was capable of executing a few hundred arithmetic operations per second.

Program Languages are Developed

In 1955 Stan Poley at the Watson Laboratory wrote the first program I had encountered that improved the machine-user interface. We later came to call these programs "killer apps". The program, called SOAP for "Symbolic Optimizing Assembly Program", allowed you to use symbolic names for computer operations and data, and the assembler optimized the location of instructions and data on the drum to improve throughput. At the time I thought, "It can't get any better than this"! SOAP ended machine language programming forever. A couple of years later, when I became the manager of the Laboratory, we entertained students from the Bronx High School of Science for a couple of hours twice a month. A 13-year-old student in the group undertook the challenge of writing an improved version of SOAP. He enthusiastically described his program to me. He had named his program "CLAP" for "Computer Language for Assembly Programming". It was my awkward duty to suggest to his teacher that his program possibly could benefit from a different name. This name was used at the time to describe an inharmonious medical problem and perhaps still is. Those students were wonders and proved to me that people at a very young age could become skilled programmers.

In about the middle 50s, John Backus from IBM came up with the idea of creating a computer language that allowed programmers to enter formulas into their code and have a compiler translate the formulas into machine language code. He needed $200,000 to produce the compiler, and an IBM Vice President thought it prudent to have John Von Neumann review the idea. On a hot summer afternoon Von Neumann appeared at the Watson Laboratory to listen to Backus. An IBMer present at the meeting reported to me that Backus gave an inspired presentation with particular emphasis on the fact that the compiler would have an optimizer that allowed it to produce code comparable in efficiency to assembly language code. It would also reduce program size and save programming time. Code efficiency was a critical issue in those days and trumped improving the programmer interface. The room was hot and conference rooms and offices were not air-conditioned in those days. My informant said that during the presentation Von Neumann struggled to keep awake. After the presentation he voiced concern that introducing floating-point arithmetic capabilities into scientific calculations could cause people to lose control of error propagation, resulting in the possibility that buildings and bridges designed with the aid of computer analysis might collapse. But he concluded that it was probably worth trying. His tepid approval caused great anxiety in the Backus team, but the project was approved and, as they say, the rest is history. Fortran resulted in a huge increase in the computer market and it was clearly a "killer app".

Replacing fixed-point arithmetic with floating-point arithmetic was also a "killer app". In writing a program with fixed word size and boundaries, the necessity to figure out where to put the decimal point in every piece of data, to get maximum precision, took enormous amounts of time. But he was right to worry about a potential problem. If in your program you subtracted two numbers of approximately equal size and either or both of these numbers contained a small error (experimental or round off), you would be left with a floating-point number that was all or almost all error. It was not uncommon for people to treat computer output as if it had been written by lightning on granite. When the IBM 704 computer was available in the late 50s at an IBM Service Bureau in Manhattan, IBM provided a small amount of free time to Columbia people. When you compiled a Fortran program on the machine, the signal that the compiler had successfully compiled your program, and that execution had begun, was the appearance of a "word overflow" error light on the console of the 704. Perhaps this was an accident or someone on the Backus team remembered that afternoon at the Watson Lab.

In those halcyon days there were several hundred users of the equipment at the Laboratory; you signed up for time a week in advance and, if you were lucky, you were allocated an hour or more each week of exclusive access to a machine. People would hang around the Laboratory hoping that someone might not show up to use their time, and if someone ran over their time, cutting into your time, the verbal exchanges could get X-rated.

On one occasion, a mathematics professor was using his hour and seemed to experience serious anxiety that the card punch on the 650 would run out of blank cards. He nervously hopped up and down, and in the interval between cards being punched he would periodically quickly raise the hopper lid and flip a few new blank cards into the punch hopper. At the end of his allotted time he pulled his output from the card punch and observed that the decaled edge of the cards seemed to be randomly distributed in every corner of the deck. He had failed to feed all the cards face down, 12s-edge-first into the punch. It was an easy mistake to make. Every tabulating machine had a different rule for inserting cards; some face up, some face down, some 12s edge first, some 9s edge first. That's why college graduates were hired to put cards into those machines. A card had a printed face with the decaled edge on the top left. The top of the card was called the 12s edge and the bottom was called the 9s edge. I thought that the Professor was going to cry. He said, "I've wasted a whole week"! I told him that his output could still be rescued because the printer didn't really care whether the cards were face up or face down and he could still print his output if he inserted the cards properly into the printer. He seemed very puzzled and held the thick deck of cards over his head turning and twisting the deck as he looked at it. He suddenly exclaimed, "That's right, the holes go all the way through"! As he had this aha moment he flipped his hand and his output flew into the air and was distributed across the computer room floor. It was now my sad duty to inform him that he had indeed wasted a week. At the time I pondered whether his observation on the holes might serve as a starting point for a new computer-based academic discipline, but then dismissed the idea.

In 1960 IBM established a Systems Research Institute in Manhattan in a building across from the United Nations for graduate-level training of its System Engineers. The first faculty, all part-time, included Fred Brooks who managed the development of the IBM 360 family of computers and its software, Ken Iverson who was the developer of the computer language APL, and me. I taught Numerical Analysis. This experience convinced me that there was an academic niche for studying computer-related issues in higher education that went well beyond the design of hardware. The first Computer Science Department established at a U.S. University was at Purdue in 1962.

So how important were computers to Columbia University in support of Research and Education in the 50s? There were a fair number of projects in the Social Sciences and a few at the Medical School that involved statistical calculations. There were a small number of projects in Astronomy, Physics, Chemistry, Engineering, Economics, and Geology, and a handful of projects from all the other disciplines at the University. Computers and computing tools were primitive and trying to analyze systems of any complexity was almost impossible. Problems involving ordinary differential equations or systems of linear equations began to come into range. Numerically solving partial differential equations except in a few rare cases was not possible. But with the ad-

vent of electronic computers at the end of the decade, there were people who were beginning to dream about understanding things like the weather, and the structure of molecules. There was a programming course at the Watson Lab taught by the legendary Eric Hankam for IBM's scientific customers that Columbia students could enroll in, and I taught a course in Numerical Analysis for the Mathematics Department. But computing instruction involved only a small number of students. This situation changed dramatically in the next decade.

Chapter 2

Computing at Columbia University in the 1960s

About 1961 IBM decided to move its education program located at the Watson Lab to upstate New York and to close the Computing Laboratory at the Watson Laboratory. It negotiated with Columbia University a plan for Columbia to build its own Computing Center. I began exploring various job opportunities at a number of places. One day I received a call from an architectural firm asserting that they were designing an underground computer center building for Columbia and they had been told that they needed to consult with me if they had any design questions. I had heard nothing about this from Columbia but told them that I would be glad to help. About a month later I received a call from a Columbia Dean saying that he would like to talk to me. At the meeting he said that after consulting with a number of people, Columbia wanted to hire me as the Director of the new Columbia Center. I agreed to serve as Acting Director for a year while I continued to explore other opportunities, some of which were very interesting.

As construction of the underground building that would house the Center started, I needed to hire a staff and prepare for the delivery of an IBM 7090 and IBM 1401. The 7090 was a computer designed for scientific calculations and in it transistors had replaced vacuum tubes. The role of the 1401 was to prepare batch input tapes for the 7090 and to print output from tapes produced by the 7090. In looking for staff there were not a lot of experienced people around. I hired a few people from the Watson Lab staff, some very bright Columbia and Barnard graduates with no computing experience, and an operator with IBM 704 experience. We studied the manuals describing the Fortran Monitor system and had help from some very good IBM system engineers. Over time we developed a world-class staff. When I became manager of the Watson Lab Computing Center in 1957, I asked Wallace Eckert, the Director of the Watson Laboratory, for advice on managing people. He said: "Just hire good people and keep out of their way". His office was in a building one block from the Computing Lab and he would appear at the Computer Lab about once a year and mumble something to the staff about our doing a good job, and disappear for another year. I tried to follow his advice on management during my entire career.

I was The Director of Computing at Columbia from 1962 to 1971. During that period equipment was upgraded about every two years. The 7090 became a 7094; a 7040 was added and connected to the 7094; a 360/50 and a 360/75 replaced the 7094-7040 system to form a 360/75 and 360/50 coupled system; and a 360/91 was added and attached to the 360/75, displacing the 360/50. Operating systems changed apace.

The 7090 was capable of executing 100,000 floating-point operations per second and had a list cost of $2.9 million. Universities, however, enjoyed a 60% IBM educational discount. The 360/91 was about 50 times faster and the list cost of a complete system ranged between $15 and $20 million. In 1965 Herb Grosch stated Grosch's law: Computer performance increases by the square of the cost. This became the basic argument for Universities to operate a large central computer. You got more bang for the buck with a faster computer. With the development of computers on a chip and microcomputers, Grosch's law was superseded about 1980 by Moore's law. Moore's law stated that the number of transistors on silicon chips doubles approximately every two years. Gordon Moore also stated this law in 1965.

During the transition from the Fortran Monitor System to IBSYS on the IBM 7094, a hacker helped us. One night a message was printed out on the console of the 7094 that said, "The phantom strikes", and all of the tape

drives were rewound past their load point and the tape ends were left unattached to their rewind reels and dangling in their columns. There was no way the hardware was supposed to let that happen.

The tape drives were attached to a channel control box, which had a wired-in program controlling the drives, and the 7094 sent instructions to the channel program directing its tape control actions. Ken Iverson had written an APL description of the channel program and apparently some student had read it. In this case a student had discovered that if a tape was at its load-point marker, and a set of cleverly designed 7094 instructions were sent to the channel-control program with precisely timed delays between those instructions, the channel program began looking for the load mark on the tape after the tape had already coasted past it. A rewind instruction then brought the tape off its rewind reel.

At the time we were running Fortran monitor batches and IBSYS batches with the operators dismounting and mounting the system's tapes between running each batch. IBSYS enabled running a number of additional applications not available under the Fortran Monitor System. With the discovery of this bug, which had now become a "feature", we were able to put both operating systems on the same tape—separated by load and end tape markers—and include IBSYS and Fortran jobs in one batch, with the system spacing over the load and end of tape marks to the system called for by a program without changing any of the instructions in either operating system. Over the years, a number of computer bugs were reclassified as features. A few years later I taught a course in operating systems for the Electrical Engineering Department and there were students in my class who were smart enough to become world-class hackers. Operating systems contained hundreds of thousands of instructions with the number increasing rapidly. Removing all vulnerabilities was impossible. There were always a few students who wanted to "make their bones", to use a Mafia term, by bringing the system down and I was teaching them what they needed to know to do it. A University Computing Center Director at a meeting once advised his fellow Directors that if they caught a good hacker, hire him or her. I found this to be very sound advice.

Computers at that time had very small main memories with access times measured in microseconds supported by secondary storage like disks, drums, and tapes with access times measured in milliseconds to seconds. In order to increase throughput, evolving operating systems tried to hold multiple programs in main memory so that when one program was retrieving data on a secondary storage device, the computer could switch to run another program in memory that was ready to run. This strategy greatly complicated operating systems.

When the IBM system 360 was introduced, its operating system OS/360 provided multiple options for multi-tasking. Unfortunately, it took a long time to get most of the bugs out of the system and University environments being very diverse, experienced many more system crashes than commercial sites. A system crash meant taking and printing a memory dump, identifying the program causing the crash in order to remove it, and restarting the batch. This process took many minutes of time. The systems group was then tasked with finding the bug and removing it. Users of the Computer Center measured service quality by turnaround time, the time between job submission and receipt of program output, and crashes were very bad for turnaround time. This was a miserable time to be managing a University Computer Center.

Users of large-scale IBM scientific computers had a user group called SHARE. In the 60s I chaired the university's SHARE group. Universities were always willing to share useful software and SHARE meetings were a place to learn about interesting applications being developed at other Universities. Technical attendees shared information on operating system bugs and the problems associated with installing updates to the operating system. These updates were frequent and sometimes affected many users, which did not make them happy. At Universities, major system upgrades were almost always scheduled for August when most faculty members were away. This had the consequence that some faculty would return in September, tanned and eager to resume their research and instruction, only to discover that, as President Nixon once said, "the information you have been given is no longer operational".

Every evening most of the SHARE attendees would convene in a large space to attend an informal event called SKIDS. At the center of the meeting space were tables full of booze and ice. Information exchanges at that event were often very useful. I quickly learned that the stress associated with the conversion to OS/360 in Industry was almost equal to that at Universities. We had less control over our users and a lot more crashes. You could approximately measure a person's stress level by how long they stayed at SKIDS. Late one evening at SKIDS, a person who managed computing at an insurance company told me that the Vice President for Marketing had a year earlier proposed to the Board that they augment the customer account number by a digit to record policy-holder characteristics that would aid marketing. When a board member asked if that would screw up data processing, the Marketing Vice President replied, "Hell, its just another hole in a card". There was also the annual Ivy League Chowder and Marching Society's meeting of Ivy League Computer Center Directors. The rate of change in membership in that Society was also a good measure of the stress. At the meetings, members of the Society would estimate how many steps ahead of the hangman they were at their institution. This generally averaged to about half a step.

In the 60s it became possible to connect printing typewriter terminals to computers, and later CRT (Cathode Ray Tube) terminals. Those terminals enabled programmers to bypass keypunches and produce, edit, and submit their programs directly to the batch and retrieve output to a terminal. These terminals could be hardwired to a computer by connecting twisted-pair wire from the terminal to the computer, or they could be connected over telephone lines using modems. The modems were acoustic couplers that you inserted your telephone receiver into and they converted an analog signal into a digital signal. It was easy enough to connect terminals at the computer center, and a terminal room was created, but challenging to hardwire terminals in remote buildings to the center. Columbia had heating tunnels bringing steam to buildings on the main campus and they contained steam pipes wrapped with an insulator that I hope was not asbestos. In any event it was reported to me that in the dead of night, armed with a flashlight, a staple gun, and a wheel of twisted-pair, unidentified people would staple twisted-pair to the back of the pipes out of sight and bring the wire to the basement of buildings where unindicted co-conspirators from a Department were waiting. This was probably a violation of some code but evidently there were people on campus who believed that if the end doesn't justify the means, what the hell does, or something like that. Several decades later it became necessary for many Universities to rewire their entire campus to accommodate computers, but the 60s were the dawn of the remote computing revolution.

Computer terminals enabled the next major advance in computer operating systems, this one called time-sharing. MIT had developed a time-sharing system in the early 1960s called CTSS that ran on a 7090 that was not capable of supporting a large number of users. Much more ambitious undertakings were started at MIT (MULTICS) and at Dartmouth (DTSS) in the middle 60s on hardware more amenable to the task. The goal was to provide a large number of users with shared terminal access to a computer with real time interaction. These systems, installed at a number of Universities, were very successful in environments where most users ran small programs. This was a characteristic of most instructional computing. But alas, in environments where research, administrative, and educational users shared a large central computer, some running very large programs, a time-shared computer could spend most of its time rolling these big programs in and out of memory from secondary storage and throughput was hurt. Throughput still trumped user ease of use almost everywhere. Columbia did not adopt time-sharing because very large research and administrative programs were a big part of the load. Computer courses used fast Fortran and PL1 compilers developed at Waterloo, Purdue, and Cornell to support instructional computing. Terminal support programs like Wilbur from Stanford enabled users that had access to terminals to enter, edit, and submit jobs and to retrieve output from a terminal, which provided some of the convenience of time-sharing without its inefficiencies.

Terminal access enabled the dawn of the age of social computing. Users were assigned a small amount of online space to store the programs that they were editing and some data. An encryption capability was provided so that sensitive information, like medical data containing patient names, could be protected. I was told that the

gay community at Columbia was using this capability to support a dating service. You "friended" someone by giving him or her the password and encryption key to the file. The encryption key was called the magic word. I was also told that the students involved in the 1969 insurrection at Columbia used this capability to coordinate their activities.

Computer Literacy

As the 60s advanced Universities began to recognize that computers and computing were an important area of instruction and research. At Columbia, a programming course that I taught in the Electrical Engineering Department attracted hundreds of students each semester. The term "Computer Literacy" became an important issue for discussion, although a lot of people didn't understand what it meant and some regarded the phrase as a linguistic barbarism. An attempt at a simple explanation of the term follows. If you look at the content of papers in academic journals, you will find that there is a hierarchy of languages used to describe the system being studied. For relatively simple systems there may be a lot of mathematics in the paper, because mathematics permits the description of systems without ambiguity. If you are mathematically illiterate, however, there are papers containing mathematics that you will not be able to understand. Papers describing systems too complex for mathematical description usually employed technical English and ordinary English to convey the points trying to be made, but ambiguity is embedded in both. Linguists point to as simple a sentence as "Mary had a little lamb". This could mean: Mary owned a little lamb, Mary ate a little lamb, or Mary had sex with a little lamb. People whose native language is English, but who are illiterate, are unable to read and understand the knowledge contained in papers written in English. The new languages in the language hierarchy enabled by computers are formal or computer languages. By combining mathematics and logic, these languages make it possible to describe complex systems without ambiguity, and they enable analyzing systems too complex for mathematical description. They also enable the capturing of intelligence that can be applied to a broad range of problems. Thus, understanding how to model systems employing a computer language is a literacy issue and an invaluable capability for any well-educated person. Farther down the language hierarchy are papers employing metaphors and poetry to describe systems as complex as humans. Maybe the hierarchy should be inverted with poetry at the top and mathematics at the bottom.

In the late 60s a faculty colleague in Electrical Engineering and I wrote a letter to the Provost at Columbia telling him why we thought it was important for Columbia to consider creating a Computer Science Department. He sent us a two-sentence response: "How can you create a science around a device? If computer science, why not X-ray science?" Columbia did not get around to creating a Computer Science Department until 1979. When I was very young, I asked my grandfather at the depth of the Great Depression if he had had a tough life. He replied: "During my life I've learned that you lose some and you lose some. You have to learn to take the bitter with the unpleasant".

Columbia University had started a Seminar Program some years earlier that involved inviting people outside of Columbia to meet with Columbia people to get a varied perspective on some subject. The seminars were recorded and the discussions transcribed with the notion that at some point a book could be created. There was one at that time dealing with criminal justice. Outside members included former criminals as well as members of the criminal justice system to get a varied perspective.

In the late 60s Ted Bashkow from Electrical Engineering and I created a Seminar on the Impact of Computers on Society. Outside members came from as far away as Brookhaven, Yorktown, and the Bell Labs. The Seminar members would meet at the bar in the Faculty Club about 6 PM and would shortly adjourn to a conference room to hear a speaker. The Faculty club had a bartender who asked me what I would like to drink the first time I showed up at the bar. The second time he simply asked: "The usual Dr. King?" After that he would prepare my drink and hand it to me as I reached the bar saying simply "Good evening Dr. King". He set a standard for

service that I have never seen equaled. About 1970, we invited the Provost to speak at the Seminar. To our great surprise, he said that he was willing to participate in a discussion on the impact of computers on Society with the members of our Seminar. He turned out to be a very charming person full of amusing anecdotes. We talked about Computers and Society but not about the subject of the earlier letter. It was the first and only time I ever met him. I came away wondering if, instead of reporting at the level of buildings and grounds, I reported at a level where I had more frequent access to him, if I could have changed his views. About 25 years after I left Columbia I was invited to speak at the Seminar. I was astonished to hear that it had survived all of the intervening years and accepted the invitation. As I approached the bar at the Faculty club at 6 PM the Bartender said: "The usual, Dr. King?" This exchange erased all of the bad memories I had of some of the events that had occurred during my Columbia years.

Speaking of bad memories, I will now turn to the subject of how computers were financed at Columbia. When the center was started, Princeton and Columbia argued that access to computers was as important to researchers as access to the library and, like the library, computer costs should be put in the indirect cost pool and not charged directly to Federal grants. They secured agreement to this practice and users of the Center were not be charged for computer time while this agreement was in effect. Once a year a Federal Auditor would show up at the center to look at time records to determine if the University was correctly including time used by researchers with research grants in its indirect cost pool. We maintained a closet with monthly computer time use records by project and jobs run. In a year's time the stack of printer output paper was about 6 feet high. When the auditor arrived we would offer to transport the stack to the conference room if that would be more convenient for his analysis. He always responded, "I don't think that that will be necessary", and left after looking at the stack for a few minutes. In fact Columbia was always totally honest in allocating computer time to the indirect cost pool.

In 1967 the Government decided that it would no longer accept inclusion of computer costs in the indirect cost pool. Research grants had to be charged directly for computer time. This change lead to the end of "civilization" as we knew it. Many researchers had small grants that could not be augmented to include their computer costs. For others it was easier to get one time money to buy a mini-computer than to get recurring funds for computer time. When computer time seemed free, other researchers did not consider the issue of whether or not a dedicated computer that they controlled would make them more productive than standing in line at the computer center. They now considered that question and frequently the answer was "yes", owning their own computer would make them more productive. Thus the "computer wars" began. The University decided that buying a departmental or project computer was forbidden unless a petitioner could prove to me that there was no way to solve their problem at the Center. This put me squarely between former faculty friends and the administration. Computer wars were common at a number of Universities. Their Computing Directors were often among the SKIDS attendees that closed the bar.

I recall a few other inharmonious events during my years at Columbia. In the first winter in the new underground computer center we discovered that the string of offices down a hallway were all in the same heating zone. At one end of the hallway was my office, which was over a loading dock. At the other end of the hallway was a conference room over a University heating plant. In order to keep the temperature in the conference room at 85 degrees, the temperature in my office had to be about 55 degrees. The staff with offices close to mine wore coats. The staff with an office close to the conference room wore tee shirts. I wrote a letter to the firm that had designed and built the center demanding that they fix this problem. I received a response telling me that they were no longer responsible for building problems.

I responded with a letter telling them that I was, at my own expense, commissioning the Long Island Casket Company to create a bronze plaque to be hung in the computer reception room. The plaque would simply state: To commemorate the nobility of their firm, and below that the date on their letter denying responsibility. I stated in the letter that when visitors, humble and distinguished, visited the center and asked what the firm

had done to merit this distinction, we would tell them. I ended the letter by stating that it was my hope that in the years ahead, when Columbia men gathered in the evening over cigars and brandy, and the conversation turned to great feats of engineering, their firm would be remembered. Shortly later I received a call from the firm telling me that they would, at their expense, fix the problem in about 3 months time. The caller then said that he hoped that I would still be there when the fix was completed. I told him that if freed from my University responsibilities, I would have plenty of time to explore whether or not the anecdotal value of this episode merited inclusion in the Talk of the Town section of the *New Yorker* magazine. He hung up. A short time later I received a communication from the Trustees telling me that they had sole approval authority over all commemorative plaques hung in University buildings. This was the only contact that I ever had with the Trustees of Columbia University.

One day the computer receptionist threw open my office door and shouted, "There is a nut in the computer room with a knife threatening the operators"! I immediately tried to reach security but their line was busy. As I kept trying, the lights on two other lines on my phone started flashing. My secretary had run to the computer room to witness what was happening. In those days there were lights for each incoming telephone line on your phone. The lights told you whether the line was free, busy, or if it was flashing someone was trying to reach you on that line. Compulsively I hit the button on one of the flashing lines and recognized the voice of the faculty member calling me. I said: "I can't talk to you now Bill, there is a nut in the computer room with a knife and I'm trying to reach security but their line is busy". His response was: "this will only take a minute".

On another occasion, the IBM 360/91 began crashing with a mean time to failure of about 30 minutes. The IBM Customer Engineers that were resident at the center claimed that it must be the software. The Columbia systems programmers claimed it must be the hardware, because they had not changed the software. This kind of finger pointing was not uncommon. After a few hours the Customer Engineers agreed that it must be the hardware. After working and watching crashes the rest of the day, the Customer Engineers escalated the problem to the Region, and a new set of Engineers arrived the following day. After these Engineers worked for a time they escalated the problem to a higher level and another new set of Engineers arrived. This set concluded that the problem was in a specific box but running their hardware diagnostics they couldn't find any problem with the box. On the third day a chauffeur driven limousine arrived carrying the designer of the box. He entered the computer room smoking a cigar and asked to see the log that the Customer Engineers kept of every hardware maintenance change. After looking at the log for a minute he pointed to a line in the log, uttered a couple of sentences to an engineer and started to leave. I shouted: "You can't leave until you're sure it's fixed"! He ignored me and departed. Five minutes later the computer was back up and running and a Customer Engineer handed me a one-inch wire. Removing the wire had solved the problem.

About 30 years later I got a call from a lady who was studying some recently declassified documents. In a document, Isador Rabi, a Nobel Prize winning physicist at Columbia, had warned the Government that computer hardware could be very unreliable and pointed to an event at Columbia in which a computer had been down for three days. She asked: "What was that all about?"

In 1969 there was a student insurrection on campus demonstrating against the Vietnam War. Students took over some buildings and the President called in the New York City police. The police wielded their batons on student heads and the consequence was that many uninvolved students, shocked by the sight of students staggering out of buildings with blood streaming out of their heads joined the insurrection and more buildings were taken over. At this point the President evidently decided that calling in the police was not a good idea and a stalemate resulted with students hanging out the windows of occupied buildings shouting defiance. I recall seeing a student standing on the lawn outside one of the buildings holding a sign that said: "Sarah, don't forget to take your pill". The Computer Center was largely locked down but I, standing at the entrance with an armed campus policeman, let in trusted users.

The FBI evidently had a wire on one of the members of the Students for a Democratic Society (SDS) and it was reported to me that at an SDS meeting a member proposed pouring iron filings into the core memory of our machines on the grounds that we were doing classified research. Fortunately for us, another high-level SDS member was a chemistry graduate student whom I permitted to spend the night in the center with a small number of other students who were trying to solve "very big problems". I was told that he vehemently asserted that he lived at the Center and we were not doing classified research. Some days earlier he had asked me why a cover had been put over a printer by a user who stood at the printer to secure his output as it was printed. I explained that the man standing at the printer was a psychiatrist from the Medical School and the output he was protecting contained transcripts of patient sessions. The psychiatrist resembled Sigmund Freud. I was informed that the student persuaded the SDS to drop the iron-filing plan.

One of the pleasures of managing computing at a University was that it provided a window into a broad range of research on very interesting things. Lunch at the Faculty Club with other faculty was often particularly interesting. On one occasion there had been a headline that day in the student newspaper that said: "Barnard girls not worth losing sleep over". That precipitated a discussion at lunch over whether or not undergraduate instruction at a men's college needed to include training on how to deal with women. One of the faculty members present said that men were untrained in issues that went well beyond how to deal with women and proposed developing a curriculum in Life Science. We agreed to meet periodically to discuss what might go into such a curriculum. One of the members, Professor Levine from Mathematical Statistics, developed an algorithm for optimally selecting a spouse. The prescription in the algorithm was that you begin by choosing a number n, which was the number of women you were prepared to court. Then you divided that number by e (one of my favorite numbers) the base of the natural logarithm system and rounded up to the next integer. You then courted n/e. However, you didn't marry any of them, but you remembered the best one. Now you continued to court until you found a better one and married her. I found that this was an excellent algorithm for selecting a new apartment to rent in New York City or for finding a building in which to put a Computer Center. It was the sole output of the effort to create a curriculum in life science.

Impact of computing technology changes

So what impact did computing technology changes have on Columbia University in the decade between the end of the 50s and the end of the 60s? The percent of faculty involved with computing grew from perhaps 2% to I would guess about 20%. All of the science departments now had some faculty whose research depended on computing. The Biology Department, relatively new to computing use, had faculty undertaking big computer projects. The use of computers in the Business School and at the Medical College had grown substantially. The availability of statistical packages caused a dramatic growth in Social Science computing. There was an occasional project in the Humanities enabled by a new programming language called Lisp. Problems that were a dream at the end of the 50s were now being tackled. They included modeling the structure of the earth using seismographic data, trying to understand climate and weather, and programs designed to understand the forces holding molecules together. Two Columbia faculty members involved with computing at that time, Martin Karplus and William Vickrey, went on to win Nobel Prizes, and if there had been a Nobel Prize in Geology, it almost certainly would have been won by Jack Oliver, who proved that the Pacific volcanic ring of fire was caused by one tectonic plate sliding under another. Computer courses had large and rapidly growing enrollments.

These changes were enabled by substantial improvement in hardware performance made possible by the transition from vacuum tubes to transistors. Computers were now capable of executing up to 10 million instructions per second. A hardware designer at this time told me that instead of resting at the end of the 6th day, he wished that God had worked on a better velocity of light. Physicists later determined that a slight change in the velocity of light would have caused the big bang to terminate in a black hole. The velocity of light created an upper limit on the speed of a computer circuit. Main memories were still small relative to the requirements of

many problems and secondary, random-access storage devices were slow and lacked capacity. Magnetic tapes were still the primary bulk storage medium. Improvements in operating systems significantly improved job throughput. The availability of terminals connected to computers enabled the eventual elimination of punched cards as the program input medium and substantially improved the human-computer interface. Institutions that could use time-sharing systems experienced significantly improved programmer productivity and quickly were able to get a majority of faculty and students involved with computing. It had begun to be clear to me that Universities running batch processing systems would have a tough time moving the percentage of faculty using computers above 20%.

There was a proliferation of new computer languages, some like Lisp directed toward a class of applications. A burning question became "would third-generation languages like Algol or PL1 replace Fortran as the work-horse language at Universities?" That question persisted as new third generation languages like Pascal appeared and the answer was always "no". A joke current at the time had a computer scientist climbing to the top of a mountain and shouting heavenward, "Will any computer language ever replace Fortran?" A voice came down from the heavens and said: "Yes, but not in your lifetime". And then the scientist asked: "Will Universities ever appreciate the importance of computing to the University?" And the voice came down from the heavens and said: "Yes, but not in my lifetime". General purpose packages such as SPSS (Statistical Package for the Social Sciences) and database management software became available that significantly improved programmer productivity in some application areas. Keeping up with new program acronyms became a real challenge. At one meeting I raised my hand and told the speaker that I did not understand his 3la's. He asked: "What's a 3la?" I replied, "a 3-letter acronym".

In 1970 the Federal Government seriously reduced funding to agencies supporting research at Universities. This produced a serious problem at Columbia because the salaries of many faculty members with Research Grants were partially funded by their Grants and many of these faculty members had tenure. One proposed solution to the resulting budget deficit was to cut the budget of administrative services at the University across the board by 10%. Another proposal from some faculty members in Arts and Sciences was to eliminate the School of Engineering. The University Board of Trustees opted for the 10% solution. One consequence of the dispute was that some members of the faculty in the School of Engineering vowed to never again speak to faculty members supporting the elimination of the School. At the 25th anniversary of the founding of the Computer Center, I met a former colleague in the Electrical Engineering Department who informed me that he had remained true to his vow. The Computer Center was an administrative service and the only way to reduce the budget was to fire staff. The staff at this time was not large and many members worked many more than the 40 hours a week that they were paid for. Woe is me!

Chapter 3

Computing at City University of New York in the 1970s

At that time City University of New York (CUNY) had just committed to open admissions and the student population in its 19 colleges was in the process of doubling to more than 250,000 students. Any New York City student who graduated from high school was guaranteed a seat in one of its Community Colleges or its Senior Colleges. One day I received a call from a CUNY Vice Chancellor telling me that he would appreciate the opportunity to talk to me. When I arrived he announced that the computing situation at the University was desperate and they needed me to fix it. He offered me an appointment as a University Dean. Once again I was offered a job without an interview. In the discussion that followed, he told me that there were no computer wars. Faculty members were free to buy any computer that their grants would fund. Second, that the highest priority at the University was instructional support, followed by enough administrative computing to keep the University afloat. Almost all of the Colleges had small computers supporting administrative computing, but a couple of new colleges would need help. There was no immediate crisis but there were clouds on the horizon. Administrative Computing reported to me at Columbia and I knew that some of those clouds could be pretty black for a rapidly expanding University.

The most attractive feature of the job offer to me was that it rescued me from my Columbia dilemma. If I took the job I could bring with me all the Columbia people that I was faced with firing. But before accepting the job offer I said that I would develop a plan and if the plan were funded I would accept his offer. I knew the computer time used by a Columbia student in a computer course, and multiplied that by a generous estimate of how many CUNY students would be computing if CUNY had a robust computer literacy program. That load alone required two large mainframes. With that capacity we could also support research and administrative computing with no problem. I prepared a budget and the Vice Chancellor explained that the funds needed would not be a problem, because the City had increased the Universities' budget substantially to support open admissions. He walked me in to talk to the Chancellor, Robert Kibbee, who assured me that he would support my plan. I agreed to accept the job. I thought about how I would tell the Vice President that I reported to at Columbia that I was leaving Columbia. One idea that crossed my mind was that I would start by telling him that I had some good news and some very good news. The good news was that he would have no problem in achieving a 10% reduction in the Computer Center budget. The very good news was that I was resigning. But I couldn't do that. I had many great memories of my time at Columbia and I was sad to leave.

I quickly learned that there was one more hurdle at CUNY. All computer acquisitions needed to be approved by someone in the Mayor's Office of the Budget, who ran a data processing operation on a small IBM 1410 computer. Above the door to an office he shared with his deputy was a sign that read: "Abandon all hope ye who enter here" (my memory may be a little vague on this point). The Vice Chancellor arranged for him and me to meet with the Mayor's Director of the Office of the Budget. The Director was a graduate of City College and a jovial fun person to deal with. He agreed that I would have no problems getting any needed acquisition approvals from the city.

My immediate problem was to hire some key staff members and to find a site for the computer equipment that I ordered a few days after I started working at CUNY. After looking at n/e possible sites for the Computer Center, I found a new building on 57th St and 11th Avenue that had vacant upper floors that were for rent. The building had large windows with a magnificent view up and down the Hudson and of Hoboken, New Jersey. After living in a cave during my Columbia years I was determined that the new Center would have windows.

We rented space on the 16th floor and the landlord took responsibility for renovating it to our specifications with the cost being included in the rent. I could not in good conscience hire senior members of the Columbia staff. That could lead to chaos at Columbia. Among the junior staff, Ira Fuchs was a big picture person who was innovative, creative, and very technologically informed. He had been the person responsible for bringing Wilbur and Orville from Stanford to Columbia. He was the most informed person at Columbia about networking problems. Connecting computers at 19 colleges, and to a huge number of terminals distributed across the 5 Boroughs of New York City, to the new CUNY Center was going to be challenging. I was confident that if anyone could solve that problem it would be Ira. He had had no management experience but that was a problem easy to solve. I could tell him to just hire good people and keep out of their way. I hired Ira as Director of the new Center. Ira recruited a number of other junior Columbia employees who were highly talented. To reach the 10% budget cut goal, Columbia did not need to fire any Computer Center employees. They did not replace me for two years. I began to believe that I had not left a vacancy. A decade later, Ira undertook the challenge of connecting terminals and computers all over the world when he started Bitnet.

I had now embarked on a new career as a University bureaucrat. As a University Dean I was a member of the Chancellor's cabinet. My office at CUNY headquarters on 80th Street in Manhattan was several doors down from the Chancellor's. At his cabinet meetings I learned about the important things happening at the University. Most of the time the Chancellor just wanted to talk about the educational issues associated with welcoming thousands of new students to the University, many of them needing remedial instruction. At the midpoint in my career at CUNY I was promoted to Vice Chancellor, and Institutional Research and Television were added to my responsibilities. The Institutional Research Office was staffed with social scientists who evaluated educational experiments and programs across the University. The Television Office provided television services across the University and supported an experiment enabling students at four Colleges to take courses offered at another college connected by television links. Now a conglomerate, I recalled a Bob Newhart line about the confidence he felt when he flew on an airline run by the Thompkins Airline and Storm Door Manufacturing Company.

The social scientists discovered, among other things, that teaching via television worked as well as teaching with all the students in one classroom. The remote classrooms had a telephone link to the instructor, and the instructor could see the student asking the question on his or her television screen. However, the cost of television links was too high to support extending this experiment. An attempt to teach students a subject with computer terminals also produced interesting results. In a mathematics course it was learned that about 75% of the students succeeded in passing the course, but about 25% hated it and did not do well. In another analysis, it was learned that female students from deeply impoverished neighborhoods did much better than males from the same neighborhood. In particular, a single parent mother on welfare did much better than her male peers with a similar high school record. By controlling for as many variables as they could think of, they concluded that this result could only be explained by motivation.

When I arrived at CUNY I recruited a very talented group of people to modernize the Universities' administrative systems. Some of the systems they created survived for more than 30 years after I left CUNY. I spent a lot of time traveling to the Colleges to talk to administrators at all levels, faculty, and occasionally to students. This meant that I spent a lot of time in New York City traffic. As a perk, I had exclusive use of a City-owned car with City license plates. Those plates enabled me to park anywhere in the city without fear of getting towed for illegal parking. On one occasion my car broke down on the Long Island Expressway. I abandoned it and another car was delivered to me. Occasionally I would visit the Computer Center, which was an oasis away from the many problems dealt with by the people at 80th Street. I discovered that the overhead associated with being a bureaucrat was very high and not a lot of fun. In particular, trying to move hundreds of faculty that didn't report to me was like trying to herd cats.

New York City Goes Broke

As the old saw goes: "Cheer up, things could be worse! So I cheered up and sure enough things got worse"! In the summer of 1975 it was discovered that the City was on the brink of defaulting on its bonds. The Federal Government and the State stepped in and imposed stiff conditions for guaranteeing the City loans to avoid default. An Emergency Financial Control Board was created by the State Legislature to review and approve every financial decision by the City. Employees of CUNY were not paid for several weeks until the State relieved the City of responsibility for funding the Senior Colleges and assumed control. CUNY had been tuition free since its creation. The State and City decided to end that practice.

In the fall of 1975 the Chancellor informed me that Abe Beame, the Mayor, had called him and that the Mayor would like to talk to me that afternoon. I traveled down to City Hall and the Mayor told me that consultants had reviewed the City's computing infrastructure and concluded that it needed a lot of fixing. In particular the computers in the City did not talk to each other, and worst of all the Budget Office computer didn't talk to the Controller's computer. The Mayor controlled the city budget and the Controller paid the bills and did the accounting. The Mayor and Controller were independently elected officials. The inability of the Budget computers to talk to the Controllers' computers was a major source of the City's financial problems. They recommended the City create an Office of Computer Plans and Controls to fix these problems. He said that the Chancellor had agreed to loan me to the City if I were willing to head this office. I assumed that I had been selected because I would be free, and of course everyone knows that you get what you pay for. After a moment or two of contemplation I decided to accept this task because it might have high anecdotal value, and it certainly did.

While Abe Beame was Mayor I reported to him. When Ed Koch succeeded him on January 1, 1976, he created an Office of Operations, reporting to him, which consisted of 30 high-level executives on loan from major New York City corporations. The lending corporation paid their salary just as CUNY paid mine. The task of this group was to look at every operation in the City for opportunities to find cost savings through improved efficiency and reorganization. Lee Oberst from AT&T was appointed as the Director of the Office. I became attached to the Office as Deputy Director. Lee was an efficiency expert. He believed that in analyzing an operation you started with the people at the bottom of the organization who were doing the work. You watched them work and measured their output and compared it to that of commercial company employees doing the same work. You also looked for ways to increase their productivity. You then worked your way up the organization toward its top management with the goal of streamlining it. The last person you talked to was the person managing the whole operation. Needless to say, this created great anxiety among City Managers at every level.

The first assignment for two members of our group was to follow garbage trucks on their collection routes. They discovered that there were some trucks that deposited almost as much garbage on the city streets as they collected. The result of their work was the reduction of staff on each truck by one, and instead of enjoying a couple of weeks off following a heavy snow storm because the streets were impassable, garbage trucks were fitted with plows and when it snowed they were out plowing their routes. A few years earlier, after a heavy snow storm, the *New York Times* interviewed a person in Queens two weeks after the storm, who said: "The only thing moving out here are the children". With 4000 garbage trucks equipped with plows that never happened again. However, for some inexplicable reason, the image of high-level executives in expensive black suits, white shirts, and expensive ties following garbage trucks and lurking in halls at night to watch janitors still causes me to chuckle. But amazing things were learned and lots of things changed.

I did not have to watch civil service programmers to know that they were not up to the task of rewriting the City's administrative systems. At CUNY I had succeeded in getting all the programmers that I hired classified as academic support staff to avoid the necessity of hiring people bound by civil service rules. During my two years at the City, all system design and programming work for the City was contracted to commercial companies. Armies of programmers across the country were hired to rewrite all the key systems and a number of

new programs that emerged from the work of the Office of Operations. We worked long days. Lee convened meetings of our group at 6 AM or 7 PM so that no one could claim a competing commitment.

After my appointment as czar of computing I began working on a plan to fix computing in New York City. Consulting the best minds at CUNY, Ira Fuchs and Mel Ferentz, a computer scientist from Brooklyn College, we worked on a plan for supporting a transaction-intensive environment. Mel and Ira and I had earlier worked together to bring Unix to CUNY. I signed the first contract exporting Unix from Bell Labs to a University. At that time the Digital Equipment Corporation (DEC) was developing an operating system that could support an extend-able bunch of Vax computers on a ring network. The system was specifically designed to support transaction-intensive environments. It looked like a winner. I developed a Vaxen plan and asked Lee to check it out with the people at Bell Labs. He reported that they loved it. I asked Lee to join me in visiting DEC to see what state the system was in. At DEC headquarters their systems and sales people told us how awesome it was. Remembering the first two years of OS/360, I then asked if the company would be prepared to guarantee that it would work reliably from day one. After some discussion the sales people decided to bring Ken Olsen, DEC CEO, into the discussion to close the deal. They huddled with Ken outside the door to our conference room and explained the deal to him. After about 5 minutes Ken opened the door, looked at us, and uttered a single word: "No"! It was back to the drawing board.

I then designed a system built around large IBM mainframes. I had no problem getting the Mayor, Controller, and City Council to approve the plan. I then did my n/e thing and found a great site for a new NYC Computer Center. The site on 10th Street had formerly been a bus terminal. It had elevators designed to transport buses from floor to floor. The city rented a floor and the Controller rented the floor above to house their computer. It was now easy for the two computers to talk to each other and they were able to share communication connections to City Departments. You could drive your car into the building and into an elevator, and the elevator could drop you and your car off at the door to your office. How's that for a world-class perk? I recruited Joe Giannotti from the CUNY Computer Center to direct the new City Center, which supported the agencies reporting to the Mayor. Joe had great political as well as technical skills. Joe hired some extraordinarily good people from Columbia and CUNY to staff the new City Center.

Two years after the City was on the verge of default it was relieved of most of the restrictions imposed by the Emergency Financial Control Board, and everything again became immersed in politics. Lee went back to AT&T and I recruited a successor czar and went back to CUNY. A few years later, as a measure of how successful and important the new Center had become to the City, Joe Giannotti was promoted to the level of a Commissioner. At that level he enjoyed a car that had a flashing red light on its roof and a siren. It was always a pleasure to travel around the city with Joe. My experience over my two years as Czar was exhausting but rich in anecdotal value. I learned that some things are only accomplish-able during a period of financial crisis.

The imposition of tuition caused a dramatic drop in enrollment at the CUNY Colleges. Budgets across the University were slashed. Even tenured professors were fired. When I returned if I asked people how things were going the response ranged from: "don't ask"! to "terrible"! to "oy vey ist mir"! The budgets of the offices reporting to me were not cut. It was good to be at the front of the trough when the slop was poured in. Everyone in the Central Office was engaged in damage control. Gradually the situation improved but those were not fun times. At the Computer Center we managed to sell an IBM mainframe and from the proceeds of the sale replace it with an Amdahl (an IBM clone) that was 30% faster. This taught me that it was wise to sell and replace computer equipment while it still had value, if you could manage that. When IBM announced the 3000 series in the late 70s we managed to replace the small very aged computers at the Colleges with 3000-series computers that were 10 times faster. As the decade of the 70s ended we began to populate the Colleges with a small number of Apple II's as the wave of the future.

It is difficult for me to assess the advances in computing during the 70s because I no longer worked in the trenches. Computers got faster and operating systems improved and computer-related instruction grew dramatically. Even Columbia had established a Computer Science Department and many Universities had computer literacy requirements. On the hardware front, the number of transistors in integrated circuits on a chip had been doubling every two years, as predicted by Gordon Moore in 1965. This was known as "Moore's law". It was now possible to put a computer on a chip. Advances in chip density would soon dramatically change computing everywhere. Three new "killer apps", email, spread sheets and word processing began to become important. Many Universities had local email systems starting in the 60s. They were important in eliminating telephone tag. But not many faculty members had terminals on their desks. In the 1970s, ARPANET proved that people would like to talk to people at other Universities, but this was not widely feasible at Universities until the emergence of Bitnet in 1981. The VisiCalc spreadsheet program for the Apple II was a clear winner at Business Schools. Wang word processors began to appear in departmental offices, but word processing became important to faculty members and students in the 1980s, when microcomputers attached to networks appeared everywhere on campuses. It was clear to me that the 1980s would be a lot more exciting than the 1970s at Universities.

Chapter 4

Computing at Cornell University from 1980 to 1987

In 1980 I received a call from Cornell University inviting me to visit the campus because they were recruiting a Director of Computing and several people had recommended me to them. CUNY had a relationship with the Mt. Sinai Medical School and there were close connections between Mt. Sinai people and Cornell Medical School people. Both Schools were in Manhattan. Through this grapevine I had earlier learned that, to use the technical term, the Cornell Medical School had been "screwed" by the Cornell Computer Center. The Cornell Medical School had been forced to move its computing off a local machine to the campus computer because the campus center was in financial trouble. It turned out that the campus computer had been unable to accommodate their load and the whole deal needed to be undone after the Medical School had sold their computer. Thus I considered Cornell as a place for which the 40-foot pole was invented, a place that you wouldn't touch with a 10-foot pole. However, it was always pleasant to visit a campus and I agreed to talk to them.

Bob Cooke, who was the head of the search committee, picked me up at the Ithaca airport. As we headed toward campus Bob explained to me that Cornell was a very complicated place with nine Schools, each with a large amount of autonomy. I told him that I had had some experience with complicated places. He then told me that the University Computing Board had persuaded the Provost to create a Decentralized Computer Support group in Computer Services to encourage and support faculty acquiring their own computer. The computer wars were over at Cornell. As we crossed a one-lane bridge entering the campus, my view of Cornell changed 180 degrees. I told Bob that I thought that microcomputers were the wave of the future, and that minicomputers and microcomputers were the only hope of greatly increasing the involvement of faculty members in computing.

During the visit I learned that high-level Cornell executives on the financial side had back loaded the amortization of the campus mainframe in the expectation that sponsored research would significantly increase as the years passed. When this didn't happen they had a problem. The campus mainframe was about 10 times faster than the Medical School computer. Someone decided that selling the Medical School computer and moving Medical School computing to the campus could solve the financing problems of the Ithaca mainframe. The campus computer had enough spare capacity to easily support a 10% increase in its load. The campus mainframe was at the airport, five miles from the campus, so the computer center staff had experience in supporting remote computing.

This solution looked as easy as shooting a pheasant that was walking toward you through a rain barrel with a rum raisin in its mouth. The move went forward without any benchmarking. Then, as they say, something funny happened. It turned out that administrative systems at the Medical Center had been written in APL square code. In APL it was possible to write dense logical code on a single line that would compile into many instructions. A special typewriter with its own unique keyboard and characters produced APL code. Bright programmers sought to get as much logic on a single line as possible, stretching the code to as close to 80 characters as possible. The challenge was to write a program with lots of long expressions. This was called "square code". Evidently the Medical Center's programs had been written by a genius. Some years earlier I had visited the IBM Yorktown Research lab and decided to drop in to chat with Ken Iverson, the author of APL, and a colleague of mine when we were both on the faculty of the IBM System Research Institute. While we chatted, someone approached Ken and asked if he could interrupt our conversation for a minute. He had a long length line of APL code that had a logic error and he needed a minute of Ken's time to figure out where. After about

10 minutes of staring at the line, Ken figured out where the problem was. The Medical Center had acquired a computer that had microcode that increased the speed of APL programs by a factor of 10. The campus mainframe was not a model that supported this microcode. The campus mainframe was brought to its knees trying to run Medical School APL programs. Apparently this episode did not make the Cornell Trustees happy and the campus was in search of new computing leadership.

I had a very pleasant day on the campus. Everyone had good words to say about the technical competence of the computer staff and several faculty members said that Douglas Van Houweling had been particularly helpful to them. Everyone thought that computing needed more financial support from the University. At the end of the day I was taken to meet Keith Kennedy, the Provost, and he indicated that he thought that the quality of computing at Cornell was very important. He walked me across the hall to meet Frank Rhodes, the President. About a week later I received a call from Keith telling me that having completed the search process, the search committee had recommended that Cornell hire me. I told him that I would think about it.

The overhead associated with being a bureaucrat had become higher at CUNY when the State took over funding the Senior Colleges. I now had to travel to Albany to clear computer acquisitions and funding that related to them. As I weighed the delight of again being able to lunch with faculty members trying to change the world and the great view of a lake from the Cornell campus against the prospect of endless drives to Albany, I decided to accept the Cornell offer if they agreed to accept a plan that I would develop. I called the Provost and told him that I needed to spend a couple of weeks on campus to develop a plan and that I would come to Cornell if my plan was accepted. I warned him that I would be unwilling to come to Cornell unless they aspired to having a first-rate computing environment and that would not be cheap. He told me that he thought that Cornell was ready to spend more money on computing and looked forward to getting my plan. He arranged to put me up in a house on campus once occupied by Cornell Presidents. It seems that the President's house had moved off campus too. During my time on campus I talked to all of the Deans and found that some were more enthusiastic about computing than others, but none were resistive. Following the wisdom of Lee Oberst's great teachings, I had hoped to talk to them last but scheduling difficulties prevented that.

In talking to the faculty I learned that Cornell had a great Computer Science Department established in 1965. Better still, the person teaching the Introductory Computer Science course was Tim Teitelbaum. Tim had been one of the people living at night in the back of the Columbia computer room. At Columbia Tim was working on a Hough Powell device that was channel connected to the center computer. The device scanned film of particle collision events in Cloud and Bubble Chambers and digitized particle tracks. I subsequently discovered that Tim deserved a great teacher award. Tim was the dynamo at Cornell behind the computer literacy efforts in the Computer Science Department. He had written a PL1 program synthesizer for a Terak microcomputer and was in the process of converting instructional computing in the department to microcomputers.

As I talked to faculty doing research I discovered that faculty running their own computers were happy with the support provided by the Decentralized Support group led by Doug Gale. A small group of faculty in the sciences led by Ken Wilson, who soon thereafter won a Nobel Prize in Physics, had cooperatively purchased a Floating Point Systems (FPS) 190L array-processing computer that was attached to a channel of the campus mainframe. The mainframe provided input and output services for this computer that was designed to support computationally intense programs. Its computational engine was as fast as the mainframe's and could sit in the Computer Center running 24/7 under the exclusive control of a few scientists. Better still, the charges to their grants for mainframe time were small because the mainframe did not need to provide a lot of cycles to keep the array processor humming along. When asked about their future computer requirements, they told me that they could use a number of Cray Supercomputers. Doug Gale remembers that the number was five. The faculty members using the campus mainframe were not happy with turnaround time, and there were communications problems exacerbated by the fact that the computer and most of the Computer Center staff were five miles from campus. Faculty members I talked with vividly remembered the attempt to run Medical

School computing at Cornell. By this time in my career I had learned that faculty were far happier and more productive if they controlled the computer resources they required. Putting them in an environment where their research was constantly interrupted by hardware, and system changes, and other inharmonious events made them less productive and unhappy. Often, a brief bit of experience in an environment they shared would teach them to avoid computing in their research and instruction if they could. There was some support for faculty not enjoying grants containing funds for computing time that was arranged in some mysterious way by Doug Van Houweling, who managed Academic Computing Support in Computer Services.

The Administrative Computing staff lived in a basement and the staff was small. There was a room with a bunch of keypunches in it at the entrance to their space. That told me almost all that I needed to know about the state of administrative computing at Cornell. Staff morale was low. A few years earlier, Cornell had experienced a budget crisis and a committee was appointed to search for savings. According to administrative computing staff members, they recommended the elimination of administrative computing. When they discovered that this would make it impossible for Cornell to pay its faculty and staff, they recommended exempting the payroll staff from their recommendation. I suspected that this story might have been a bit exaggerated but the result of the committee report was downsizing the staff by 40%. The staff that survived could tell me the names of everyone on that committee, and some of the people who joined the staff after this event had learned their names too. Thus their legacy was preserved.

Computer Services probably had the best Virtual Machine (VM) systems' programming staff in the country. There were two major IBM operating systems at that time and the one with the most interactive features was VM. Bob Cowles, who led this group, was a product of the Cornell Computer Science Department as were a number of other people in Computer Services. He had written a VM scheduler program that was widely used and in my view clearly superior to the IBM-provided VM scheduler. Dick Cogger, who was innovative and creative in supporting the campus connections to a computer five miles away, led the networking group. Overall, the staff was small but contained some very talented people. Its morale was low because many people on campus blamed them for the Medical School fiasco. However, with proper support they were clearly up to the task of rapidly improving the quality of computing at Cornell.

In assessing the computer problems at Cornell, I concluded that Cornell was a "poster child" for the shortsightedness of Federal Funding Agencies in not allowing the inclusion of computing charges to sponsored research projects in the indirect cost pool. As a consequence of that decision, many Universities had organized Computer Centers as enterprises, which were expected to earn enough revenue to cover their expenses, except for as small a contribution as possible from general revenues. All computer time was billed. At Cornell, the vicissitudes of depending on the sale of time to sponsored research caused the Computer Center to scramble to sell time to outside groups. When this failed and a budget deficit resulted, often heads rolled. Across the country uncountable amounts of computer time were wasted because sponsored research projects, unable to get sufficient funding in their grants to pay for computer time, were denied access to the campus computer. In addition, in determining the hourly rate for time, all Computer Center costs were bundled into a cost pool to be divided by the time available. This included every employee of the Center. Service Bureaus were selling time for a lower rate because the only staff they needed to support their enterprise consisted of a few operators and a few sales and technical support people. In my view, not using the Library Model to finance Computer Centers was a national tragedy.

I prepared a plan for improving computing at Cornell. First of all, computer equipment would have to be amortized at a rate that reflected its resale value. Ideally the University would get first in line as a new generation of computers was introduced and the equipment it replaced would be sold while it still had value. The goal was to insure that the University maintained state-of-the-art computers. Secondly, the University would have to create a number of computing support rooms containing networked terminals and microcomputers distributed across the campus and into the dormitories. The number of these computer-access rooms would have to be

increased as an aggressive effort to infuse computing throughout the University curriculum increased support requirements. The third goal of the plan was to insure that every Cornell graduate had received training that enabled them to understand the role of computing in their discipline. Thus the notion of "computer literacy" went well beyond teaching them how to program and supporting the requirements of the Computer Science Department.

To infuse computing broadly into the curriculum, the University had to get as many faculty members as possible computer literate. This could not happen if the primary computer resource on campus was a batch processing system. The University would have to find a way to support faculty access to computing in every discipline, including disciplines that did not enjoy grants enabling them to buy computer time. An important goal was to find ways in which computing could be used to improve faculty research productivity. Fourth, the University needed to rewrite most of its administrative computing systems to provide online access to data to every office requiring that access. This would require significant expansion of the administrative computing staff. Finally, I had two nonnegotiable demands. The first was that I report to the Provost and the second was that the Computer Center would be moved back to the campus from the airport as soon as suitable space could be found. Ideally this space would be close to the center of the campus.

I presented my plan to the Provost and he said that the University would review my plan and that I would hear from him shortly. A few days later I received a call from the Provost at 8 AM telling me that the University would accept my plan and I agreed to accept their job offer. I felt a lot of loyalty to CUNY and I was sad to leave. When I resigned, I recommended that Ira Fuchs replace me as Vice Chancellor. The Chancellor agreed and it was a wise decision. A year later Ira started Bitnet, one of the transformative events in the evolution of technology. Bitnet proved that networking was important to Universities and laid the groundwork for the creation of NSFnet, which laid the groundwork for creating the Internet.

I was the Vice Provost for Computer Services at Cornell from 1980 until 1987. When I left Cornell, the Computer Services budget had grown by a factor of three during that period. Cornell had over 600 microcomputers in rooms across the campus and in the dormitories. Many students were buying their own microcomputers at a store run by Computer Services. An important goal was to make available microcomputers costing no more than $1000. We fell slightly short of reaching that goal due to a high New York State sales tax. Cornell had become one of five National Supercomputer Centers. All the buildings on campus including the dormitories had been rewired. The wiring plant included fiber, coaxial cables, and twisted pair. The cable supported access to television in student rooms via a connection to the local cable company. The computer center had moved from the airport to a renovated building near the center of the campus fulfilling a promise made to me by the Provost when I accepted his job offer. A building had been constructed adjoining the Computer Center to house a University-owned telephone system. With the University, rather than the telephone company, owning the wire that reached every faculty and staff member's desk, a large number of faculty and staff had a microcomputer on their desk connected to the University network.

The University mainframe had been replaced with the latest large IBM mainframe. When it was acquired we spoke of it as the last University mainframe. It was augmented with several medium-size computers that served the computing needs of a subset of computer users. One of them was shared with The Cornell Institute for Social and Economic Research (CISER). This significantly improved the productivity of the CISER members. The administrative systems of the University had largely been rewritten to support online access to a central database and the University had an online Library catalog system.

As an historical note, in a speech at Cornell a number of years later, Bill Gates said that a Microsoft employee, who was a Cornell graduate, returned from a campus visit in 1986 and announced "Cornell is wired". This caused the company to start developing a networking capability in its operating system. As another historical note, the last campus mainframe was retired in the summer of 2013. When I left Cornell, I judged its comput-

ing environment to be first-rate. This was enabled by unwavering support from Provost Keith Kennedy and his successor Bob Barker; support at critical times from a faculty advisory committee initially chaired by the Dean of the School of Engineering Tom Everhart, followed by Bob Cooke; and an extraordinarily talented and creative staff.

Major Computing Events at Cornell

John Rudan has written a *History of Computing at Cornell*, which is in the Internet-First University Press (IFUP) section of the Cornell eCommons. You can find his history at: http://hdl.handle.net/1813/82 . As a side note, Bob Cooke and I created the IFUP as part of a project that we started in 2003. A section of this Rudan history covers the details of staff organization and hardware acquisitions, and catalogs major events during the very busy period that I was at Cornell. In this memoir I will only discuss the background of major events that moved computing at Cornell, sprinkled with some anecdotes designed to convey the issues faced by people involved in University computing at that time.

In 1981 Ira Fuchs at CUNY persuaded Grey Freeman at Yale to share the cost of connecting their computers via a 9600 baud leased line to create a store-and-forward email network that Ira hoped would one day connect every scholar in the world to every other scholar. The network protocol they used was developed at IBM and an IBM group had created a small email network using this protocol operating under the radar at IBM. Ira had a friend at IBM from the time that they both were employed at the Columbia Computer Center who informed him of this development. Ira called this network Bitnet and as the network came up Ira began calling me regularly to find out when Cornell would connect. Because of a peculiarity in the telephone tariff rate for leased lines, it was cheaper to lease an interstate line than an intrastate line. When Penn State University connected to Bitnet, Cornell connected to Penn State becoming the 6th University to connect to Bitnet. Shortly thereafter, the University of Toronto connected to Cornell becoming the first international member of Bitnet. About this time Ira reinterpreted the name to mean: "Because It's Time" network. Soon California institutions connected to each other and to CUNY. The network began to grow rapidly. University College Dublin was the first European institution to connect. The Director of Computing at the College, Dennis Jennings, later became responsible in the NSF for networking Universities to the National Supercomputer Centers to create NSFnet, which evolved into the Internet. Bitnet spread across the US, South America, and Europe. There was even a connection in the Soviet Union. When the Tokyo Institute of Technology leased a line to CUNY, it spread to Asia. A networking history document in Japan displays the first message from a Professor at the University to the US. It was a message to me from a Professor that I helped when he computed at the Watson Laboratory in the 50s. The capability of Bitnet was soon augmented to support email lists and instant messaging.

In 1987 people on Bitnet received a message from a student at Clausthal University of Technology in Germany asking them to execute an embedded program to print out a Christmas tree and receive a Christmas greeting. The program was able to retrieve your Bitnet address file and forward this message to everyone on your list. The resulting traffic created by this "worm" brought down Bitnet and all the commercial networks connected to Bitnet. In 1988 a Cornell computer science student distributed the first Internet worm. The student had received his undergraduate degree at Harvard. People at Cornell observed that students who got their training in ethics at Harvard and their training in Computer Science at Cornell could be very dangerous.

At Cornell, aside from the Cornell Computer staff, who benefited from connections with their technical counterparts at other Universities in order to share solutions to common problems, the early users of Bitnet were students who quickly discovered Bitnet and were used to hanging out in network-connected terminal rooms that were beginning to spread across campus. Faculty members soon discovered that they could undertake collaborative research with faculty members at other Universities without having to play telephone tag or use snail mail to exchange papers. Long distance calls were expensive in those days. Bitnet was a free service. It cost the University $5K per year for the leased line to Penn State. Faculty members began to include their Bitnet address

on their business cards. When the campus was rewired to bring a network connection to every desk, a major incentive for getting a microcomputer on their desk was that it enabled email connections to their colleagues at Cornell and around the world using Bitnet. A study conducted in the late 80s showed a large increase in the number of papers authored in Journals by people from more than one University. This increase was attributed to Bitnet.

An early effort involved finding ways for faculty members without access to computing funds to compute. We created a program called "Windfall", which enabled them to use weekend computing time on the mainframe that would otherwise be unused and wasted. Since all computing time had to be billed, we created a very low rate for this time and the Provost provided funds that were turned into "funny money" that was allocated to faculty to pay for this time. It was called "funny money" because it could only be used to buy computer time on weekends. After we started this program, a faculty member's wife complained to me that she rarely saw her husband on weekends because he had become a wind-faller.

Cornell had a very creative Controller named Jack Ostram. In 1962 Jack was at Princeton and the person Columbia financial people worked with to get computer costs into the indirect cost pool for a few years. Jack reduced the cost of computing time by stripping out of the cost base, used to determine the hourly rate, everything except the direct cost of providing cycles. All other costs were put into something he called "the glob", and these costs went into the University's indirect cost pool.

In late 1983 Dan'l Lewin showed up on the Cornell campus with a small metal box handcuffed to his wrist. In the box was a Macintosh computer. He demonstrated the little jewel to me and I immediately knew that the computing world had changed forever. It had a graphical user interface with icons on a bit-mapped screen and a mouse as a pointing device. It came with only two applications, MacWrite and MacPaint, but Lewin explained that many more were under development. It introduced a new word into the computing vocabulary, WYSIWYG, "what you see is what you get". Lewin, who was a Cornell graduate, told me that Apple was organizing a group called the "Apple University Consortium", which would consist of 24 of the country's most prestigious Colleges and Universities. The group would be advisory to the company and enjoy a special discount on Macs. He invited Cornell to join and without hesitation I accepted. Later that evening I called Ira Fuchs and told him that he should try to get CUNY into the Consortium. The next day he called me to report that he had talked his way in.

Shortly thereafter, a complimentary Macintosh was delivered to Cornell. I decided to demonstrate it to faculty and computer staff people present at our 5 PM Friday Wassail. People played with it and were able to navigate without any instruction. There was high excitement and glee among everyone who watched. At some point I decided to show them the 3.5-inch floppy that contained its operating system. An attempt to eject the disk from the Mac failed. I picked up my phone and dialed Steve Jobs's telephone number. This number had been provided to Consortium members. To my surprise and delight Steve answered the phone. I explained the problem. He told me that there was a small hole under the disk and that I should insert a paper clip wire into that hole and the disk should come out. He said that he would stay on the phone until I reported success. I inserted the wire and the disk popped out. The people there cheered. I got back on the phone and told Steve that it worked just like magic and thanked him. At that moment I recalled the IBM hardware designer who left the Columbia Computer Center without waiting to see that his advice had solved a problem that had kept the 360/91 in the center down for 3 days.

I started the Friday wassail tradition while I was at Columbia in an effort to gather data that might be useful in a stress management component in a Life Science course. The wassail was presided over by a non-management employee called the "Chief Degenerate". On Friday at 5 PM I set a modest amount of booze out on a table in my office and when this was done it was the Chief Degenerate's responsibility to stand at my office door and blow a loud whistle called the wassail whistle. Anyone hearing the whistle was welcome to come and have a drink. On

one occasion at CUNY, a person from Brooklyn College arrived a little late. He explained that he was on a subway platform in Brooklyn waiting for a train to Coney Island when he heard the whistle and decided to cross the platform and catch a train to Manhattan instead. The Chief Degenerate had life tenure unless promoted to management and had sole authority to throw someone out of wassail or even ban someone from wassail. It was his responsibility to enforce my only rule: everyone had to make his or her own drink. No one was permitted to make a drink for someone else. I only remember one incident in which the Chief Degenerate exercised his awesome power. Someone on the Center staff started to bad mouth a staff member who wasn't there and the Chief Degenerate told the person that he would be ejected unless he changed the subject. I suspect that that person would not have evoked this reaction if he had been badmouthing a member of the administration instead of a member of the staff. I learned a lot from the wide range of people who attended Wassail. It was an event at which every level in the organization could meet. On one occasion at CUNY, I had both the Chancellor and the building janitor at Wassail. The Chancellor asked the janitor how many keys he had on his belt. On another occasion a fellow Vice Chancellor arrived with a distinguished poet. The poet asked me who my favorite poet was? I replied: "Robert Service". He replied: "that's not poetry, its doggerel", and set down his drink and left. I wondered what his response would have been if I had replied: "Robert Frost, Dylan Thomas, or Omar Khayyam, among my other favorite poets. But it's hard to beat an opening line like: 'A bunch of the boys were whooping it up in the Malamute saloon' ".

Over time we populated a lot of rooms across the campus with Macintoshes. One of the sites was in the Undergraduate Library. We persuaded the Undergraduate Librarian to make a small room in the Library available for Macs. We installed about 15 of them one evening and the next morning I got a call from a Librarian. She said: "You better get over here in a hurry. There's a serious problem in the Mac room". When I got there I discovered 15 students working on Macs and another group sitting on every free space on the floor waiting for access to a Mac. The Librarian thought that they were occupying the floor space as part of a protest movement. After that, every time I saw her we started to laugh.

During the years that I was at Cornell we had a lot of small renovation projects as we acquired space for computers. One very big effort involved renovating an old building on campus and moving the computers and staff from the airport to campus. John Rudan managed these projects in addition to managing the Systems and Operations staff. One day a fellow Vice Provost told me that she had been waiting in a queue for 6 months to get her office painted and wondered how I managed to repeatedly jump everyone in the queue. She suspected skullduggery. On an occasion somewhat later I decided to accompany John on a visit to an office to reinforce the urgency of their providing a critical service for us promptly. John entered the office and greeted the 3 people in the office by name. He not only knew their names but the names of their wives and children. He asked every one of them for a family update. He then got around to discussing the reason we were there. They promised to start the next day. I never had to open my mouth.

Cornell had discovered that most students arriving at the University had poor writing skills, so every freshman was required to take a writing course. As students started to use word processors, someone raised the issue of whether or not a word processor improved or diminished a student's writing quality, so an experiment was initiated. A randomly selected group of students were allowed to use a word processor while another randomly selected group was required to use a conventional typewriter. It was discovered that papers from the conventional typewriter group were clearly superior to those of students using word processors. In looking into the issue, it was discovered that students using a typewriter spent a lot more time on their papers than the word processor group. Because their papers required many retyping's, they typically started work on their papers on Monday if the paper was due on Friday. The word-processing group had adopted a "just in time" model and often started work on the paper late on Thursday. The typewriter group thought about their paper as they walked across campus. After lots of discussion, it was decided that it was futile to ban the use of word processors. Instead the course was modified to require revision of papers that were less than perfect. A paper was marked up

and returned to the student for correction. Students were required to submit their papers electronically rather than on paper to facilitate the instructor's revision process.

One of my first tasks at Cornell was to rebuild the Administrative Data Processing Group and to undertake an effort to put all of its important shared data online and accessible from terminals. To manage this effort I hired an extraordinarily good person named Russ Vaught who at the time was Director of the State University of New York at Binghamton's Computer Center. Building information systems at a University is very hard. While working in the Office of Operations in New York City, we started at the bottom of an organization and worked our way up with the goal of saving money by automating procedures and streamlining the management structure. At Universities, collegiality required starting with the Vice President responsible for an administrative service and working your way down. It seemed almost never possible to acquire a commercially available system, even if good ones were available, because they made the paper flow in the office different. From the perspective of office managers, the goal was to add new services that enhanced the value of their office to the University. These new services almost always required adding staff in the office. A new system was always expensive to build and the end product usually improved service but cost more to operate than the system it replaced. That was true at most major Universities.

A major long-term controversy at Universities was the question of whether or not both academic and administrative computing should report to a University Chief Information Officer (CIO). In the 70s, as computing became more important to Universities, Computer Center Directors began to be appointed above the level of buildings and grounds, and there was a strong trend to merge academic and administrative computing under a CIO. My own view was that if they shared a large campus mainframe there were advantages to a unified management structure. However, like research faculty, administrative offices always seemed happier if they controlled their own hardware.

I learned a few things from Russ. He was constantly working on staff development. He explained to me that it was important to send his staff to opportunities for training and education. He would say: "We've got to invest in our seed corn." He quickly developed a very talented staff. In the early 80s, mainframe computers had far less CPU power, memory, and storage than a desktop PC today and the whole university shared that computer. Thus it was easy to undertake projects beyond the capability of the hardware. There were examples of large administrative systems started at a University by someone who could be characterized as possessing infinite optimism bounded only by unlimited self-confidence. These projects always ended badly. Russ carefully analyzed the requirements of every system that he was asked to implement before he committed to develop it. Russ tested new technologies in something called his sandbox. One of the technologies in his sandbox was object-oriented computing. This technology later became very important to supporting the software maintenance of microcomputers on campus. Academic computing was always a lot more fun than administrative computing and when Russ left Cornell it was to become Director of Academic Computing at Penn State.

An exception to the usual decision to build our own was the acquisition of a commercially available online Library Catalog access system. A committee was created, co-chaired by Jan Olsen, Director of the Mann Library at Cornell, and me to select a system. We visited a number of Universities that had installed a system or were in the process of installing a system. At one University that we visited, academic computing reported to the Director of the Libraries. I asked the Director when she thought the library system they were implementing would be up and running. She replied: "When it's ready. It's taken 200 years for this library to get where it is, and if it takes another 200 years to implement this system, that's how long it will take".

As we started our work, I was asked to explain the goals of the effort to a faculty Library Advisory Committee. I decided to start my presentation with a bit of humor. I quoted a Harvard Librarian who said: "The goal of a library automation system is to keep the faculty out of the Library". No one laughed. I got that "what planet is this guy from" look. As this memoir is being written, Cornell has recently closed libraries in the Physics and

Chemistry Departments and the School of Engineering because of lack of use. Paul Ginsparg, who created the arXiv system making preprints of papers available in mathematics and the sciences on the Internet, was recently quoted as saying: "today, its Google or forget it". ArXiv has catalyzed advances in Physics by making preprints of papers available months before they appear in Journals. Access to information on the Internet from the comfort of their offices and home is apparently now keeping the faculty out of the library.

In 1984 the IBM President visited a number of Universities and was shocked to discover that their student computer rooms were predominantly stocked with Macintoshes rather than PCs. So that same year IBM started an Advanced Education Project (AEP) with major grants to 19 Universities including Cornell. With this grant Cornell acquired more than 500 PCs over the next three years. Under the terms of the grant, most of these computers were given to faculty members in return for a commitment to develop instructional material for one of their courses that used a PC. We named this project "Project Ezra" after Ezra Cornell, the founder of the University, and Cecilia Cowles managed it. We always included the Cornell Medical School in programs directed toward faculty, and they were recipients of some of these PCs. We quickly succeeded in establishing excellent relations with the Medical School. Faculty members developing software were mostly quite senior with tenure. It was generally discovered that it was a big mistake for junior faculty members to interrupt their research careers by spending time writing instructional software. The Ezra staff members at Cornell were fierce competitors and Cornell always had more projects selected for presentation at AEP meetings than other AEP Universities. I was fond of quoting the first law of the Yukon to staff members: "The scenery changes only for the lead dog; if you're second or third the view can be fairly monotonous".

As Cornell faculty members began developing software for microcomputers a controversy developed about who owned the software. In the mainframe days, commercially marketable software produced by University employees was treated as "work-for-hire" and copyrighted in the name of the University. Historically, software that had the biggest impact was "open source" software that was freely distributed and collaboratively developed by its users. Berkeley Unix was a prime example. As faculty members at Cornell began to develop software for microcomputers, the Cornell Trustees insisted that that software be copyrighted in the name of the University. Faculty members argued that software for microcomputers should follow the paradigm of faculty produced books, which were copyrighted in the name of the author. As the controversy raged, faculty-members took their efforts off campus and a number of privately incorporated companies were created to develop and market their software efforts. After I left Cornell the Trustees realized that the effort to own microcomputer software was hopeless and gave up.

One of the super techs on the Cornell staff when I arrived was Steve Worona. I offered him a management position but he said that he did not want to manage people so he reported to me. Steve had written the campus email system some years earlier and he maintained it and extended it. While I was at Cornell, Steve created two new systems called CUINFO and "Dear Uncle Ezra," both of which still survive. CUINFO contained general information about Cornell of interest to faculty and students. "Dear Uncle Ezra" was a section in CUINFO that allowed students to submit questions to an anonymous Assistant Dean of students. Stripped of names, some of these questions and their answers were published in "Dear Uncle Ezra". Sometimes the questions indicated that the student submitting the question was deeply depressed and possibly suicidal. In that case the Assistant Dean took immediate action to help the student. In other cases the answer to the question required some research and consultation with experts before an answer was published. One such question was: Where's the best place in Ithaca to get "soul food".

In the middle 80s we created a project called "Broad Jump". The notion was that every "knowledge worker" needed a computer on his or her desk. The goal of the project was to put a computing device on the desk of every high-level administrative officer at the University. Steve Worona and Cecilia Cowles were assigned the task of interviewing University Officers to determine their computing requirements. They started with the President, Frank Rhodes. They began their presentation to Frank by saying: "Every knowledge worker needs a computing

device on their desk". Frank stopped them right there. He said: "I'm not a knowledge worker, I'm a wisdom worker". Undeterred, they next interviewed the Provost. Shortly after their visit to Keith Kennedy, the Provost called me somewhat upset. He reported that after talking to him Steve turned to Cecelia and said: "I think that he just needs a dumb terminal, don't you?" They had better luck lower in the administrative hierarchy.

Cornell Becomes a National Supercomputer Center

In 1982 Ken Wilson won the Nobel Prize in Physics. He had long been part of an effort to convince the government to support supercomputing. With the Nobel Prize he became a leader of that effort. In 1985 the NSF announced a competition to create 5 National Supercomputing Centers at Universities. In the middle of the competition period I was asked to talk at an IBM-sponsored conference in New Haven about the AEP-funded "Project Ezra" at Cornell. As the conference ended I was approached by Carl Ledbetter who asked if I had time to discuss something that he had been musing about. My flight back to Ithaca had been delayed by weather and I had plenty of time. I had met Carl earlier when Columbia University acquired a 360/91. Carl had been involved in IBM's entry into the supercomputer market with the 360/90 series. After selling only 18 machines, IBM had abandoned the supercomputer market.

Carl wondered if Cornell and IBM could partner in creating an application to be one of the National Supercomputer Centers. The notion was to channel-connect a number of FPS (Floating Point Systems) array processors to the largest IBM mainframe to create a computer in the supercomputer range. A chemist at IBM had already shown that this was possible. In the 50s at the Watson Lab, I had written a paper that showed that if you numerically solved partial differential equations using finite difference elements, and if you selected the right coordinate system, almost all of the partial differential equations of classical physics were amenable to massively parallel computation. In fact the only hope for solving problems like understanding weather was effectively using a massively parallel computer. The problems that Ken Wilson was interested in solving were also amenable to massively parallel computation. Wilson was an ardent fan of parallel computing. At the time Cornell had two FPS array processors attached to its mainframe, so we knew how to do that. Carl and I agreed on an equitable distribution of labor. He would try to convince IBM to donate all of the IBM equipment required to create this supercomputer, and I would try to convince Cornell to submit a proposal to the NSF employing this architecture. Parallel computing was the wave of the future and this proposal would be the only parallel computer in the competition. Carl's parting words were: "When I bring this proposal to IBM management they will probably say: "Carl, how long have you worked for IBM, not counting the rest of today?" The next morning I brought this proposal to Ken Wilson and the Provost, Bob Barker, and they were both enthusiastic. I reported to Carl that half our work was done.

We quickly developed an NSF proposal that included the donation of IBM's latest and greatest and sent it to IBM. As the deadline for proposal submission approached all we could learn was that IBM had not yet made a decision. Three days from the deadline we informed IBM that we needed to know one way or the other immediately. Shortly thereafter, we received notification that if we were prepared to meet with Ralph Gomory, IBM Director of Research, at their Yorktown Laboratory that afternoon, we would get a decision. Barker chartered a flight to Yorktown and he, Wilson and I set off. On the flight I prepared some flip charts using a magic marker on overhead projector foil. Fortunately the flight was smooth so the result was legible. The foils extolled the virtues of parallel computing as the wave of the future and pointed out that our collaboration would be the only parallel computer in the competition. I finished with a statement that the benefit to IBM was that they would have a seat at the table where supercomputer applications and the technology necessary to solve "big important problems" were discussed.

I made my presentation to a group of about 12 people in a conference room. It quickly became apparent that aside from Gomory, none of the other IBMers present had heard of this proposal. They were all high-level managers at the laboratory. Carl Ledbetter stood in the back of the room not looking too confident. After I finished

my presentation, Ralph asked the people sitting around the table what they thought. One person stated that IBM had gotten out of the supercomputer business and he couldn't understand why IBM was interested. Other people around the table thought it might be an idea worth trying, with a few having no opinion. One person observed that participation could result in serious embarrassment for the company. Perhaps he was referring to an incident a short time earlier. In response to a comment from some IBM executive that IBM was not interested in supercomputers because the market was too thin, Wilson had told a joke that got national coverage. The story in the joke was that serious traffic congestion had developed on the only bridge crossing the Hudson River at Poughkeepsie. This problem precipitated serious discussion on the question of whether or not a bigger bridge was needed and consultants from IBM were hired to produce a recommendation. They reported that a bigger bridge was not needed because after careful watching, they had not detected a single person swimming across the Hudson at the bridge site. At the time of the joke's appearance in the press it was reported to me that some people at IBM were not amused. After everyone sitting around the table had voiced his opinion, Ralph looked up and said: "Alright, we'll do it; you will have a letter of commitment from IBM tomorrow morning". In anticipation of running a supercomputer at Cornell, Ken Wilson had created a Theory Center. The next afternoon a member of the Theory Center staff flew to Washington to hand deliver our proposal to the NSF.

Our next problem was to prepare for the NSF's site visit committee. We had a dry run. It was immediately apparent to me that no one, including Wilson, had prepared. After a presentation by an astronomer on the faculty, I asked: "Would you rather have a Cray?" He immediately replied: "Yes". I thought that it was all up the smokestack. On the day the site visit committee arrived every presenter was eloquent, particularly Wilson. Wilson had invited a parallel-computing expert to testify. He informed the committee that failure to advance parallel computing could set back the evolution of computers and result in a tragically wasted opportunity. The committee asked very few questions. Their smiling faces told me that we had won. The other winners were from Princeton, Illinois, Pittsburgh, and the University of California at San Diego.

The last-minute decision by IBM to support the joint proposal had made it impossible for them to create a contract with Cornell specifying what Cornell would do for them in return for their investment. They now wondered what I meant in Yorktown when I said that they would have a seat at the table. They proposed a contract specifying a number of deliverables. Most of the deliverables involved access to faculty members, but it was not possible for Cornell to impose any requirements on the faculty using the supercomputer. When stripped away there was very little left. On the signing of an agreement by Cornell and IBM's top corporate attorney, he said to Wilson, Barker, and I: "Gentlemen, you have given us the sleeves to your vest".

The multiprocessor system initially provided by IBM to the Cornell Center was quickly upgraded with a "vector" processing capability that significantly enhanced performance. The leader of the group at IBM that developed this capability was Carl Ledbetter. I now began to understand why Carl was musing in New Haven. Over time, IBM contributed equipment with a list value of many millions of dollars to the Cornell Center. Some years later an IBM president told me that the addition of vectors had enabled them to sell a lot of computers that they otherwise would have been unable to sell, and that their investment in supercomputing had been profitable to the company.

NSFnet is Created

The first issue faced by the 5 winners of the National Supercomputer Center competition was how to connect supercomputer users who were distributed at Universities all over the country to the Centers. The NSF hired Dennis Jennings, the Director of Computing at University College Dublin, and the first European College to connect to Bitnet, to manage that effort. Fortunately, Dennis had been born in England and had an English passport, which made him eligible to serve. Dennis was thoroughly Irish with a big smile and had he been born in Ireland he would not have been eligible.

A major issue in creating this network was the communication protocol to be used by the network. There were 3 possibilities. There was an International standard called OSI (Open Systems Interconnection) that was working its way through a committee dominated by hardware vendors at the pace of a pig through a python with, in my view, the same final output. There was DECnet, a proprietary protocol developed by the Digital Equipment Corporation and used by a large number of physicists connected to ESnet (Energy Sciences Network). ESnet was funded by the Department of Energy and connected Universities to National Laboratories, and other research organizations. Physicists were likely to be big early users of the Supercomputer Centers. Finally there was TCP/IP (Transport Control Protocol/Internet Protocol), a protocol developed for ARPANET. TCP/IP broke messages into packets and the messages were reassembled at the destination computer. Packets in the same message could take a different path from origin to destination. This design reflected the fact that computers and routers were unreliable and could become unavailable in the network at random times. In this case each message packet could find an alternative route if one was available. By 1985 ARPANET was no longer operational but the protocol had survived. As a graduate student at the University of California Berkeley, Bill Joy had incorporated TCP/IP into the BSD (Berkeley Standard Distribution) version of Unix. This version of Unix, called Berkeley Unix, was widely used by Computer Science Departments. In 1982 Bill Joy was a co-founder of the Sun Computer Systems Company. TCP/IP was an "open standard" whose future evolution could be controlled by its users. Ever mindful of Lenin's advice: "influence is good, control is better", the University community strongly favored TCP/IP.

Ken Wilson's wife was manager of the UNIX Support Group in Computer Services and an ardent supporter of TCP/IP, so I had no doubt where Wilson stood on this issue. I wondered if Wilson would be willing to go to the wall with the physics community if they organized in support of DECnet. I quickly got my answer. Dennis Jennings reported that at an NSF Science Advisory Committee meeting, Wilson had taken off his shoe and banging it on the table shouted: "TCP/IP, TCP/IP, TCP/IP"! Dennis reported that, aside from him, no one else at the meeting had the foggiest idea of what he was talking about. This story got broad currency and we never heard a peep from the physics community about DECnet. Some years later, at an anniversary of the start of NSFnet, Wilson, Jennings, and I were on a panel of pioneers to talk about the early days. Jennings brought up the shoe-banging episode and Wilson confessed that at the time he didn't have the foggiest idea either.

In the fall of 1985 Jennings convened a meeting of the Directors of the supercomputer centers at the National Center for Atmospheric research (NCAR) in Boulder, Colorado, to discuss the creation of a network. Sitting around an outdoors table we architected a 3-tier network. At the first level was a multi-connected network connecting the Supercomputer Centers to each other. This was called the "backbone network". At the next level were regional networks connecting Universities in a region to one of the supercomputer centers on the backbone. At the highest level was a local campus network connecting users at a University to a regional network. When we finished the design of this network, Dennis asked: "and the protocol?" Wilson and I shouted: "TCP/IP", and with a broad Irish grin Dennis said: "Done!" Dennis then told us that he was allocating $250K to the funding of the backbone. He stated that he had not gotten permission to allocate that amount but that at the NSF it was often easier to get forgiveness than permission. The next day Cornell ordered 56 kilobit-per-second lines to form the backbone, and the University of Illinois took responsibility for acquiring the routers for the network. Thus NSFnet was born. It began to be referred to as the NRN, the National Research Network.

The next day I invited the Directors of Computing Centers at Research Universities in New York to a meeting at Cornell scheduled for two weeks later to discuss the formation of a New York State Regional Network. At this meeting Richard Mandelbaum from the University of Rochester and I agreed to seek Federal and State funds to support the creation of New York State Education and Research Network (NYSERnet). We quickly secured State and Federal support to launch it. The next problem was to secure continuing funds. At the same time, Regional Networks were being formed across the country with many of the major movers and shakers coming from the Bitnet community. Doug Gale who had been Director of the Decentralized Computer Support Group at Cornell had moved to the University of Nebraska as Director of Computing. He organized the creation of

MIDnet serving Arkansas, Iowa, Kansas, Missouri, Nebraska, Oklahoma, and South Dakota. Building NSFnet was going to cost a lot of money, and we needed the Federal Government to step up to that challenge.

EDUCOM was an academic computing organization that had been involved in pushing networking since its founding in 1964. In 1984, Ira Fuchs got a grant from IBM to create a Bitnet Information Center. Among other activities this Center managed the creation of routing tables, which needed to be continuously updated as the network grew. He decided to locate this center at EDUCOM in Princeton. As a result of its previous history and the association with Bitnet, EDUCOM meetings were the principal forum at which networking issues were discussed.

At an EDUCOM meeting in 1986, Doug Van Houweling approached me with an idea. Doug had been Director of Academic Computing when I came to Cornell in 1980. In 1981 Doug was appointed CIO at Carnegie Mellon. Doug was a unique talent and his departure was a great loss to Cornell. Doug, who had a Ph.D. in Political Science, told me that we needed to organize an effort to convince Congress to fund the expansion of NSFnet to every College and University in the country. To accomplish this we needed to have an active program of educating Congressmen on the importance of this effort. Not-for-Profit organizations like EDUCOM could not "lobby," but they could "educate". He suggested that he and I jointly convene a group of CIOs with the hope that we could create a group of Universities that would fund this effort. We were jointly able to put together a meeting of about 12 CIOs. At this meeting we were able to convince 5 Universities to pony up $25,000 each to hire someone to lead this effort. There were a few more Universities whose CIOs said they would probably join this effort but they needed to check with their University before committing. By 1988 this group had 40 members and the annual dues were set at $5000. We decided to name the group The Networking and Telecommunications Task Force (NTTF). It was chaired by Douglas Van Houweling. With the money we had raised at that meeting we could start recruiting someone to lead this effort. Ed Shaw from Stanford told us that the perfect person to lead this effort was Mike Roberts, who was Stanford's telecommunications guru. We authorized Ed to try to convince Mike to take the job. He shortly reported that he had accomplished this mission. We got agreement from EDUCOM to embed Mike in their organization, and Mike reported to EDUCOM's Princeton office. As he arrived the President of EDUCOM resigned and Mike was appointed Acting President, setting back our effort until EDUCOM could recruit a new President.

In the middle 80s, developments at MIT and Carnegie Mellon had the promise of changing the world. "Project Andrew" at Carnegie Mellon and the "Athena Project" at MIT were developing Distributed Computing Systems that were designed to support an integrated computing environment across the campus. People began talking about 3M workstations on every desk. A "3M workstation" could execute a million instructions per second, had a million-byte memory and a million-pixel display. The expectation was that in the near future most people would personally control all of the computing support they required.

Over the period that I was at Cornell the most important thing that changed was the attitude of people on campus toward computing. After the middle 80s when I had lunch in the faculty club, it was not unusual to hear fragments of conversations at tables around me about computing. Students were buying large numbers of computers and it became common to see computers on faculty members' desks. The Computer Services staff felt needed and appreciated. This was in stark contrast to the mood on campus when I arrived in 1980.

Bill Arms chaired a search committee to select a new EDUCOM President. In the summer of 1987 he told me that the EDUCOM Board had decided to offer the job to me. I was happy at Cornell, high above Cayuga's Waters. However, the challenge of advancing networking to connect every scholar in the world to every other scholar was irresistible. A few years earlier I had seen a lady on the beach at Cape Cod. She wore a tee shirt and on the back of the tee shirt it said: "If we can put a man on the moon, why don't we put all of them there?" Unlike that grand technological challenge, connecting all the scholars in the world might just be attainable. I decided to accept the offer.

Chapter 5

EDUCOM Activities from 1987 to 1992

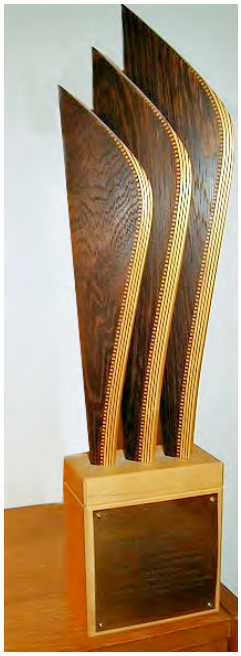
In 1987 EDUCOM was a hand-to-mouth organization. It had been living rent-free in space at the Educational Testing Service (ETS) provided by Henry Chauncy, one of the founders of EDUCOM. Henry, also the founder of ETS, had retired and under new leadership ETS needed the space that EDUCOM occupied. As I arrived Mike Roberts, the Acting President, was in the process of moving EDUCOM to rented space in Princeton. EDUCOM had virtually no surplus and had a small line of credit at a local bank. However, membership was on the rise and it was enjoying some overhead from the IBM grant supporting the Bitnet Information Center. At the next Board meeting it was decided to relocate EDUCOM to Washington, DC, when our one-year lease expired.

We found space in a building at 16th and L in DC and were now positioned to let Mike Roberts and the NTTF educate Congressmen. The NTTF became highly influential as the only organized networking group on the ground in DC representing Higher Education. On L Street we were one block from K Street where all of the lobbyists hung out. Far enough away to avoid the stigma of being confused with people who were lobbyists rather than educators, but perhaps close enough to learn a few tricks. Actually the only thing I learned from them is that the Barbecue Industry was responsible for moving daylight saving time up a month.

EDUCOM membership and EDUCOM conference attendance began to grow rapidly. In 1988 EDUCOM created an annual Net Conference in Washington, DC. If you were interested in networking, or the development of instructional software for microcomputers, the EDUCOM conferences, were where the action was. Higher education had become a big market for microcomputers and software. Vendors jostled each other for prime space in the vendor product display area at the EDUCOM conferences, and IBM and Apple tried to outdo each other in hosting EDUCOM attendees. For a period it became possible to attend an EDUCOM meeting without paying for a meal if you were adept at getting invited to vendor presentations. By the end of 1988 EDUCOM was no longer a hand-to-mouth organization.

The EDUCOM/NCRIPTAL Outstanding Software Awards

EDUCOM began pursuing two major thrusts in support of academic computing. One thrust, led by Steve Gilbert, mobilized the talents of Directors of Academic Computing at Universities in order to identify and promote quality software developed at their institution that improved education. In 1987 EDUCOM collaborated with the National Center for Research to Improve Post Secondary Education (NCRIPTAL) at the University of Michigan to create the EDUCOM/NCRIPTAL outstanding software awards program. A national review panel judged software submissions on their importance to the undergraduate curriculum, factual accuracy, and comprehensiveness. This effort was supported by the Annenberg CPB project, NeXT, Apple, and IBM. In 1987 there were 139 submissions and 27 awards for outstanding software. The number of submissions grew as the annual



EDUCOM/NCRIPTAL 1989 Trophy for Best Engineering Software

awards program continued. One winner from Cornell was Bob Cooke and his team for a finite elements program. This program allowed faculty to assign students a homework problem that would have generated a PhD a few years earlier. In 1991 Joe Wyatt, the Chancellor of Vanderbilt University, challenged EDUCOM members to produce and identify 100 outstanding software success stories. Judith Boettcher from Penn State chaired this national effort. In 1993 she edited a book titled: *101 Success Stories*.

In 1991 IBM started an annual awards program recognizing an individual for outstanding contributions to advancing computer-related education. This program was managed by EDUCOM and called the "Lou Robinson Award", named to honor a long-time IBM Director of Education who had been a tireless and inspirational speaker at Colleges and Universities on the subject of computer education. Along with a cash award came responsibility to speak at the EDUCOM conference in the year of the award. The first award winner was John Kemeny, the President at Dartmouth, who with Tom Kurtz developed the Basic computer language and started one of the first time-sharing efforts at a College. Kemeny was determined to see that wherever computing education was going, Dartmouth would get there first.

Kemeny was a chain smoker and airline regulations that prohibited smoking made it impossible for him to take a long flight to the next EDUCOM Conference to speak, so I went to Dartmouth and we taped a video interview. An edited version of several hours of interview was shown at the EDUCOM conference and was a real winner. Kemeny explained to me that cigarette companies had provided free cigarettes to soldiers during World War II, as their contribution to the war effort, and he returned from the war with a severe addiction problem. We talked about a lot of things unrelated to computing in education. He had been chairman of a commission investigating the Three Mile Island nuclear power plant accident in 1979 and we talked about that. You can watch this video at: http://www.youtube.com/watch?v=HHi3VFOL-AI

The next year's winner was Seymour Papert. Seymour, from the MIT Media Laboratory, had developed the programming language Logo that enabled very young children to start programming by controlling a robot called a turtle.

In 1990, the Association of Research Libraries (ARL), EDUCOM, and CAUSE joined together to form The Coalition for Networked Information (CNI) in order to create a collaborative project focused on networking that would integrate the interests of academic and research libraries (ARL) and computing in higher education. The first Executive Director was Paul Peters. The three sponsoring organizations sought to broaden their respective community's thinking to encompass digital content and advanced applications to create, share, disseminate, and analyze such content in the service of research and education. CNI holds an annual conference in Washington and is still going strong under the leadership of Clifford Lynch who succeeded Paul Evans as Executive Director.

EDUCOM had a consulting service that provided consulting on computing issues to Colleges and Universities by members from similar institutions that enjoyed high-quality computing. I joined a number of these groups. I learned that leadership at the top level of the University was a key element in creating a quality computing environment. At one College we ended our visit by interviewing the President. He said that he thought computing was a distraction to learning and was planning to create computer free zones on campus.

The NREN Becomes the Internet

The second major effort at EDUCOM was building a worldwide scholarly network. Mike Roberts and the NTTF led this effort. Over a period of time the NTTF developed 4 goals:

1. Connect every scholar in the world to every other scholar, surmounting the barriers of space and time to scholarly collaboration. (This goal came out of Bitnet.)

2. Enable access from the network to all scholarly information worth sharing.

3. Build a knowledge-management system on the network that enables scholars to surf through this sea of knowledge in a simple and intuitive way.

4. Enable building knowledge on a new, dynamic, multimedia foundation in addition to building it on static information in print format.

There was a fifth goal that never got any traction: To enable enable students to find instruction any time, any place, on any subject. Some people found this goal threatening to the future of Universities that still employ batch-processing systems to educate students.

When EDUCOM moved to Washington we discovered two staunch allies in Congress, Al Gore in the Senate and George Brown Jr. in the house. While Al Gore's father was in the Senate, he had sponsored the bill creating the National Highway System, and Al was eager to create a National Information Superhighway based on the NREN. As early as 1986 Al Gore began to extol the importance of high-speed networking to the future of the country. I believe that he has gotten nowhere near the credit he deserves for creating the Internet. NSFnet was now referred to as "The National Research and Education Network" (NREN). The NTTF had easily persuaded Stephen Wolff, who succeeded Dennis Jennings as the NSF person responsible for networking, to allow education as an acceptable use on NSFnet and the NRN became the NREN.

The goal of the NTTF was to create a network connecting scholars. Al Gore's goal was to create a network connecting everyone. That worried me until I remembered what Mae West had once said: "too much of a good thing can be simply wonderful!" and suddenly connecting everyone seemed like a good thing. But the NREN was our sandbox and letting everyone play in that box was frightening. The NTTF threw in with Al, and Al created a bill called "The High Performance Computing Act of 1991". There were three things that we could do to help Al: we could provide technical help with the bill's contents; we could provide experts to testify before congressional committees; and we could mobilize University Presidents to call their Senators and Congressmen to tell them how important the NREN was to their University. Our primary interface in Gore's office was Mike Nelson. Mike was a geologist by background, and we were fond of saying that Mike had come to Washington to learn the difference between a rock and a hard place.

The bill was enacted on December 9, 1991. In signing the bill, President George H. W. Bush predicted that the Act would help "...unlock the secrets of DNA, open up foreign markets to free trade, and [establish] a promise

The Gore bill passes: 1991

of cooperation between government, academia, and industry". All those things certainly happened, probably on a scale far greater than he could have imagined. On the afternoon of the passage of the bill, I was invited to a small celebration in Gore's office and I have a picture of Al at that party with an unindicted co-conspirator.

In 1993 Al Gore became the Vice President, and in April the NSF announced that the NREN would become privatized in April 1995 to become the "Internet". On hearing that news, it struck me like a bolt out of the blue: This could lead to the end of "civilization" as we knew it!

I talked to Glenn Ricart about creating a small group that would contain some CIOs and networking superstars to try to create a virtual academic network on top of the Internet. We created a group of 25 people plus a representative each from EDUCOM, Internet II, CNI, and Cause. Glenn suggested calling this group the "Stone Soup Group" after an old folk story. It subsequently evolved into the Common Solutions Group (CSG) and is still going strong. Its email list provides me with my current window into networking and academic computing issues at Universities.

In the late 80s the international standard OSI protocol began to emerge and some European institutions, particularly in Germany, began to employ it rather than TCP/IP. This required a protocol conversion gateway between TCP/IP and OSI networks. TCP/IP was a lighter, more nimble protocol and had by this time developed huge momentum. In the United States there was only one TCP/IP holdout and that was the Library of Congress. The person running computing at the Library was Henriette Avram. Henriette had developed the MARC card catalog format, the international standard for bibliographic and holdings information that was the format used at most libraries. She was a legend in the library community and a staunch believer in International Standards since she had developed one. In order to put the Library of Congress catalog on the Internet, the University of Maryland created a PC-based gateway between the NREN and the library's internal network.

I talked to Henriette a number of times, trying to persuade her to convert to TCP/IP but she seemed unmovable. It was always fun to talk to Henriette, particularly about the halcyon days when she was a 701 programmer. One day someone from Berkeley told me that students there had developed a highly successful persuasion technique. It was called: "talking a person to death". You would mobilize students to massively attempt to persuade the person by any means at hand. I started sending people to talk to Henriette but not many could spare the time. I informed Henriette of my strategy and she was amused. She said that she always enjoyed talking to computer people. In a conversation one day with the NSF Director of CISE, the Directorate for Computing, Information Science and Engineering, he mentioned that he was visiting the Library of Congress the following day. I suggested that he drop in for a conversation with Henriette if he had time. The next day, Henriette called me in a highly agitated state. She asked: "Who was that man?" I asked: "What man?" She said that a man had barged into her office and shouted: "Who the hell do you think you are?" He then ordered her to convert to TCP/IP and stalked out. I told Henriette that I would "call off the dogs". In 1992 an IBM Vice President was the keynote speaker at the Net 92 conference in Washington. He announced that IBM was abandoning support of the OSI protocol in favor of TCP/IP. At the end of his talk Henriette walked up to me and said: "I give up. He's bigger than me." I replied: "Henriette, can I buy you a drink?"

In 1992 Bob Kahn, Juergen Harms, and I signed papers of incorporation creating the Internet Society as a non-profit organization, and we appointed a founding Board of Trustees with international representation. The Internet Society was created to support the growth and technical evolution of the Internet as a worldwide educational and research infrastructure. The Society internationalized the Internet evolution effort. On the following page is a picture of the founding board holding a copy of the Society's charter.

You can learn about the contributions of these Internet pioneers by going to Google. The Society is still going strong.

The Internet Society was started in 1992

The Founding Board: In the front row from left to right are Ira Fuchs, Ken King, Bob Kahn, Juergen Harms, and Anthony Rutkowski. In the second row are Lawrence Landweber, Hideo Aiso, Lyman Chapin, Kees Neggers, Tomaz Kalin, Vinton Cerf, Froda Greisen, Geoff Huston, and Michael Roberts

The University and Research communities created new Internet applications in the early 90s. Among the most significant were web browsers at CERN and the University of Illinois, arXiv at Los Alamos, the Internet Gopher at the University of Minnesota, CU-SeeMe at Cornell, and the handle system for digital library objects at CNRI. In particular, web browsers caused an explosive growth in Internet users.

I believe that Universities have never been given proper credit for their role in creating the Internet. By the time NSFnet was started in 1985, networking was already a global University phenomenon. It was the zeal of the University computing community fired by academic goals that fueled the explosive growth of networking. Universities were the laboratory in which Internet technology and applications were built and scaled up to work with millions of interconnected people. University people provided most of the creative energy and did almost all of the work in creating the NREN that evolved into the Internet. Universities trained the people that brought the Internet into the commercial world. The role of ARPANET has been well documented but the role of Universities has not. It is my hope that the Memoirs in this incremental book will redress that problem[1].

1 An oral presentation of portions of this Memoir were presented in a lecture on February 17, 2011, for the Cornell Association of Professors Emeriti: "The Origin and History of the Internet, a lecture by Ken King" at http://ecommons.library.cornell.edu/handle/1813/22368

Chapter 6

Looking back

A lot of progress has been made on The NTTF goals:

1. Connect every scholar in the world to every other scholar, surmounting the barriers of space and time to scholarly collaboration. (This goal came out of Bitnet.)

2. Enable access from the network to all scholarly information worth sharing.

3. Build a knowledge-management system on the network that enables scholars to surf through this sea of knowledge in a simple and intuitive way.

4. Enable building knowledge on a new, dynamic, multimedia foundation in addition to building it on static information in print format.

I think that we can check goal 1 off as having been accomplished. In the 80s the challenge was to connect every scholar in the world to every other scholar. Today the challenge is to connect every person in the world to every other person.

Progress has been made on goal 2, but a lot of work remains. Some breakthroughs have been achieved toward open publication: major milestones include the preprint archive ArXiv maintained by the Cornell Library and PLOS (The Public Library of Science) that distributes open-access peer-reviewed articles on the Internet. Clearly information "wants to be free". It is the only non-depletable commodity. The more it is used, the more valuable it becomes.

In the early 2000s Bob Cooke and I began talking about problems associated with expanding the amount of information available on the Internet. Bob was concerned because libraries were beginning to drop journal subscriptions because of their rapidly increasing cost. The journal publication system involved the Government and Universities paying for research, but the scholars were publishing in journals that required the scholar to assign the copyright to the journal. The journals then sold their content back to libraries at ever-rising (greater than CPI) cost. University Presses were unable to publish scholarly work in some disciplines because the market for books in the discipline was too thin. Electronic publication of books was now feasible, but Universities were continuing to buy miles of books every year. Finding a mechanism to distribute and pay for electronic versions of books consistent with copyright laws was and still is a major hurdle to their replacing physical books. Electronic books are content-separate from a physical object that restricts usage to one person at a time. One copy of an eBook on the Internet or in the cloud could serve every University in the world. An eBook can contain multimedia material and is closer to software, videos, and music than a physical book. A site licensing mechanism could solve this problem. Just keeping physical books on a shelf in a library costs something like $6 per year. Something like 90% of the books acquired by a research library are never circulated. The paradigm for book acquisition is "just in case", rather than "just in time". Just freeing the money spent on shelving and processing miles of physical books every year could pay for a lot of pay-per-view access to electronic content. There seemed to us to be something very wrong with the current system. I was additionally concerned because I hoped to check off NTTF Goal 2 in my lifetime.

Bob was Dean of the University Faculty at Cornell at the time. We concluded that the solution to a lot of these problems was promoting Internet publishing of all scholarly work as a first choice if it were economically feasible. Bob believed that if every University assumed responsibility for the publication of the scholarly work of its own faculty, and published it in open-access mode on the Internet, they could save money.

In addition, Internet publication would vastly increase exposure to scholarly ideas produced at their institution.

We succeeded in getting a 3-year grant from the Atlantic Philanthropies Foundation to pursue the economics of Internet publication in the hope of finding a solution that would result in the Internet providing access to all scholarly information worth sharing. With funds from the grant we joined the DSpace Consortium that was supporting the development of software at MIT enabling Universities to archive information on a server whose content could be distributed on the Internet. We also acquired a server managed by the Cornell Library that DSpace required to hold its archive. At the same time we created the Internet-First University Press (IFUP) as the vehicle enabling us to experiment with Internet publication. With excellent support from the Cornell Library we discovered that lots of valuable scholarly information could be published at very low cost. We published a number of out-of-print books. One by Jack Oliver, entitled *The Incomplete Guide to the Art of Discovery* has had more than 100,000 downloads. We've published books that couldn't be published by a commercial press because the market was too thin. We've video taped Symposia and lectures by distinguished faculty members and published them on the Internet. We've put online all issues of the Cornell Alumni magazine, faculty memorial statements, and the minutes of the Faculty Senate. We have put Departmental histories online and are in the process of trying to videotape an interview with every emeritus faculty member covering their career. We have succeeded in getting most of Cornell theses and dissertations online. The DSpace archive has become the Cornell eCommons, and DSpace has merged with a Cornell-produced archive system called Fedora to create DuraSpace. You can find material published by the IFUP[2] at: http://ecommons.library.cornell.edu/handle/1813/62

From this project we have learned that it is very inexpensive to scan and OCR (optical character recognition) printed documents and to videotape lectures, interviews, and other events. The University and faculty own the copyright to lots of material of scholarly value and this material can be put on the Internet with a modest effort. Publishers are often willing to return the copyright of an out-of-print book to a faculty member. This was the case with Jack Oliver's books. If every University undertook responsibility for publishing the scholarly material in its archives that it or its faculty owns, the result would be of enormous value to the world.

Search engines like Google have enabled progress on goal 3. Artificial intelligence based systems like the IBM Watson program promise significant progress in the future. A lot of progress has been made on goal 4, enabled by inexpensive video-cams and editing and other image and sound processing software. Words like "visualization", animation, and "virtual reality" are used to describe progress. When I taught an introductory Computer Science course, I would tell my students: if you can see it and it's there, it's real; if you can see it and it's not there, it's virtual; if you can't see it and it's there, it's transparent; and if you can't see it and it's not there, it's gone. So much progress on electronic display of information has been made that the future of hard cover books and print documents like newspapers are now threatened.

Attaining "computer literacy" and infusing computing into the curriculum were huge challenges from the 50s to the 90s. Today many students graduate from high school with the skills that we struggled to impart and support during that period. Enrollments in introductory computer science courses have dropped as a conse-

2 See also: "Compressed version of the Final Project Report to Atlantic Philanthropies" at http://ecommons.library.cornell.edu/handle/1813/3459 and
"The First Ten Years of the Internet-First University Press and CAPE's Histories and Biographies Project" at http://ecommons.library.cornell.edu/handle/1813/36253

quence. Nearly every student shows up at college with a computer, frequently more than one, and they carry it around with them. The computer in their pocket is as fast as the fastest supercomputer in the mid-80s and it has a larger memory and more random access storage. It connects them to friends around the world via instant messaging and email. With Skype they can even see and talk to people around the world. It connects them to vastly more information on the Internet than they could have found in all the libraries in the world in the 90s, and almost every document is accessible with a full text index.

Research in almost all disciplines has become computer-dependent even in the Humanities. Faculty members in most disciplines control all the computing power they require. In the sciences, social sciences, and engineering the curriculum has been transformed by access to "application packages". Progress in the structure and methodology of instruction has been slower to change. In the 90s people were fond of pointing out that if a physician slept for a hundred years, and if when he woke up he visited a hospital, he would feel like he had been dropped into a planet in some other galaxy. If a professor slept for 100 years and walked into a classroom he would still find a blackboard and chalk and feel right at home. Today a screen often replaces the chalkboard and a PowerPoint presentation replaces chalk on blackboard but not a lot more has changed in many disciplines. Video-taping a lecture to a class with a large number of students and putting it online probably has more value to students than requiring them to sit in a classroom with little or no opportunity to ask questions. Long ago, educators discovered that interactive education is more effective than passive education, but most of education is still based on a sage on a stage.

Today's supercomputers are massively parallel with more than 3 million computing cores. They are more than 10-million-times faster than a supercomputer in the 80s. In signing the Gore bill in 1991 President George H. W. Bush opined that computers would one day be able to decode DNA, which was at the time one of the grand challenges of science. Today DNA decoding has opened up many new fields of research. Recently I read that falcons were genetically similar to parrots and woodpeckers, and the apple is a member of the rose family. Development of software has become a cottage industry with new apps proliferating at a rapid rate. Universities are no longer the source of most educational software.

Computer service organizations have gotten smaller and are again largely enterprise units. Their heads are now in the "Cloud". If a University can find a computer service like email in the Cloud, it is cheaper to contract for it than to support a locally developed system. CIOs at Universities no longer have the financial wiggle room to invest in new ideas. The only large-scale computing project designed to change the world that Universities are currently involved in is Internet II. This attempt to create the next generation Internet is a collaborative effort involving industry and many Universities.

In the 90s Universities were measured by the number of books in their library and big Universities added miles of books every year. Today JStor provides network access to back issues of more than 2000 journals with full text search. Miles of library bookshelves are no longer needed. Electronic books are cheaper than hard cover books and come with full-text search capability, and sometimes with multimedia elements, but using them presents copyright problems that have not yet been solved. Since Gutenberg invented the movable type printing press in 1450, great Universities were built around great libraries. Today the Internet provides access to information found via full-text searches that vastly exceeds the locally stored information in any University library; but University libraries still largely measure themselves by the number of books on their shelves.

Chapter 7

The Future Lies Ahead

Super computers 1000 times faster than today's fastest computer are on the drawing board using current chip technology. Far faster quantum computers are a gleam in computer designers' eyes. The consequences of this increase in capability on everything boggles the imagination. Technology allowing people to talk to their computer with artificial intelligence assistance in locating information is just around the corner. Artificial intelligence assistance in many areas, even in grading and editing papers, is being developed. Networks connecting homes will be supported by fiber optics cables capable of delivering data at rates more than 1000 times faster than current home data rates.

The impact of emerging technology on Universities is likely to be profound. Current education delivery systems are labor intensive and consequently increase in cost faster than the inflation rate. Tuition increases are often at twice the rate of inflation. Nothing can continue to grow exponentially without a day of reckoning. At many Colleges, adjunct faculty members that earn small salaries without a benefits package are now teaching courses. A recent development is the emergence of Massive Open Online Courses (MOOCs). MOOCs frequently enjoy enrollments in the thousands. A Faculty member teaching one of these courses gets more feedback from students in one semester than he or she would get in a lifetime of teaching in a classroom. This feedback enables the course and its delivery system to improve over time. The MOOC courses are available anytime, any place, to any student. This is the goal that I failed to get included in the NTTF list in the 80s. If great Universities can cooperate to create great courses, why shouldn't they all use those courses? Colleges are beginning to give credit to students taking these courses. I expect this trend to grow rapidly.

Current educational systems are largely batch-processing systems. When they become distributed systems, students will have courses available to them wherever they are that far outnumber the courses available in a College today, as well as access to the Internet library. Virtual Laboratories will replace real ones. If you have a question in the middle of a lecture, you will pause the lecture and ask Professor Siri for an answer. You won't even have to raise your hand. It doesn't get any better than that. However, this can lead to reducing the social interactions between students and the mentoring relationship of students with faculty members that is important in education. Someone has pointed out that the invention of the printing press freed the monks from their transcriptional responsibilities and they went off and created the wine industry. Freeing many faculty members from their teaching responsibilities will have unpredictable consequences for Universities and the world. When the NREN was privatized to form the Internet, I worried that this could lead to the end of "civilization" as we knew it. It has certainly changed almost everything.

03

GALE

The Early Years of Academic Computing:
A Memoir by Douglas S. Gale

This is from a collection of reflections/memoirs concerning the early years of academic computing, emphasizing the period when Cornell developed its own decentralized computing environments and networking became a national strategic goal.

Initial Release: April 2016 (Increment 01)
Second Release: April 2017 (Increments 02-04)

Published by The Internet-First University Press
Copy editor and proofreader: Dianne Ferriss
The entire incremental book is openly available at http://hdl.handle.net/1813/36810

Books and Articles Collection – http://ecommons.library.cornell.edu/handle/1813/63
The Internet-First University Press – http://ecommons.library.cornell.edu/handle/1813/62

C. Memoir by Douglas S. Gale

Preface and Introduction

Bill Arms and Ken King have chronicled much of the early history of academic computing. Accordingly, I have tried to focus my contribution to *The Early Years of Academic Computing* on topics less covered by King and Arms. My contribution is organized into four chronological periods, roughly spanning the last 50 years of the 20th century, and each has an organizational theme: career transitions, the microcomputer revolution, the networking revolution, and the changing role of academic computing and networking.

The theme of the first period -- career transitions -- will consider where the early practitioners of academic computing came from. Computer Science was first recognized as an academic discipline in 1962, and the early years of academic computing were led by people who had received their academic training in other disciplines. Many, if not most, came from the community of computer users and early adopters. The first period was characterized by the emergence of computing, both academic and administrative, as something of importance in higher education. Computing centers were created and awareness of their importance began moving up the administrative hierarchy.

The second period will focus on the emergence of decentralized computing driven by the advent of small, inexpensive, and powerful microcomputers. This paradigm shift involved more than hardware and software. It was a paradigm shift in how the technology was managed, organized, and supported. Higher education played a key, and largely unrecognized, roll in this transition.

The third period will focus on the contributions of higher education to the evolution and widespread use of computer networks. Much as the "printing" revolution was based on the technology of movable type and mass-produced paper, future generations will probably not differentiate between the technologies of computing and networking.

The fourth period will focus on how the relative roles of academic computing, administrative computing, and networking changed in the final decade of the century in response to the microcomputer, distributed computing, and the Internet. In particular, this period marks the relative decrease in the role of centralized academic computing and an increase in the importance of administrative computing.

The third and fourth periods were also characterized by the growing recognition that computing and networking were strategic to the role and mission of higher education.

Contents *(The headings below are linked to the pages.)*

Chapter 1

Career Transitions

Mille viae ducunt homines per saecula Romam (A thousand roads lead men forever to Rome)

Although computer science was identified as an academic discipline in 1962, at Purdue University, in the early years the ranks of academic computing practitioners were largely filled by individuals who received their formal training in other disciplines. What disciplines? What was their background? Why were they drawn to computing? How did the transition occur?

This section will follow the author's transition from a student, to a scientist and user of computer technology, and to an information technologist. My experience in conducting oral history interviews for the Internet Legacy Institute (www.internetlegacyinstitute.org) indicates that my experience, from user to practitioner, was typical of the career transitions of many of the early academic computing leaders. Hopefully, it will provide some perspective on why and how so many educators and researchers were drawn to this new discipline.

Computing for a Kid in the Fifties

For kids with technical interests, growing up in the 1950s was much different than for one growing up in the 21st century. Computers were outside our range of experiences. If we had heard of them at all, they were regarded as something used by businesses to sort punch cards or by research laboratories to calculate new artillery tables. But they had little relevance to what we did on a day-to-day basis. Our heroes, Buzz Cory (Space Patrol: http://users.bestweb.net/~foosie/spacepat.htm), and Tom Corbett (http://en.wikipedia.org/wiki/Tom_Corbett,_Space_Cadet), as well as older heroes such as the Lone Ranger and Superman, didn't need or use computers to explore the universe and bring evildoers to justice. Computers weren't referenced in either our textbooks or our comic books.

Mechanical calculators, which relied on ingenious combinations of gears to perform simple arithmetic calculations, were developed as early as the 17th century (http://en.wikipedia.org/wiki/Mechanical_calculator). But the complexity of their construction and resulting cost limited their use. If a family had a computer at all, it was likely a cheap slide rule or simple mechanical adder/subtracter.

The Boy Mechanic *Photograph by Doug Gale*

So what did a technically oriented kid do growing up in the '50s? We experimented with things, such as disassembling fireworks to gather the gunpowder in order to make skyrockets and small cannons. My friends and I built a rocket ship, which was basically a collection of old junk and anything that had a switch, and reenacted our heroes' adventures. As we got older, the switches started to become functional and many of us became ham radio operators. My call letters were K0HKY. We made things. I spent many hours as a kid pouring over a copy to the "Boy Mechanic" that my father gave me for Christmas in 1953.

It covered everything from building a boat or a working telephone to equipping a home chemistry lab, complete with dangerous and unstable items.

This hands-on, hardware-oriented approach to technology influenced many of our later careers in computing.

Computing for a Student in the Early Sixties

As a physic's major at the University of Kansas in the early '60s I, like all my classmates, did computations either by hand or with a slide rule. In my case, it was a cheap bamboo slide rule that my uncle won in a poker game. It wasn't until I went to graduate school that I could afford a Pickett, all metal slide rule.

The two indispensible tools of a science or engineering student were a slide rule and the CRC Standard Mathematical Tables, which contained tables of logarithmic and trigonometric functions, as well as other useful data.

Slide Rule and CRC Tables
Photograph by Doug Gale

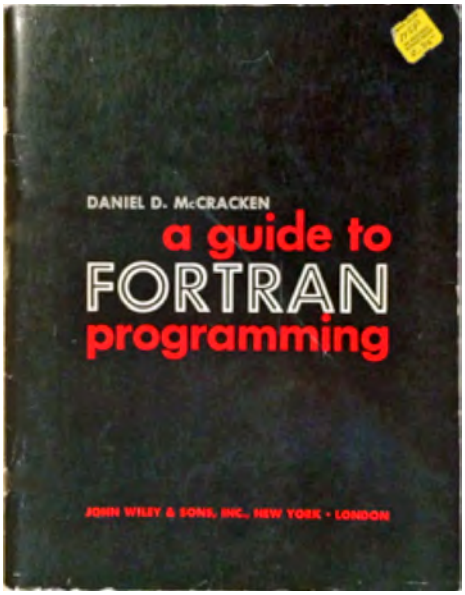
I was introduced to computers in 1963 when my roommate, an Electrical Engineering major, was taking a computer-programming course using the university's General Electric mainframe. It was a perfect match. He was looking for real problems to program, and I was trying to find a way to dress up a mundane compound pendulum laboratory assignment. Using numerical techniques to eliminate the small-angle approximation was just the ticket. I did the experiment and the mathematics, and he wrote the computer code. One of the key observations I made was that my roommate spent a great deal of time writing computer programs; so much so that he added an extra semester to his undergraduate career.

Although most students using computers came from the physical sciences, mathematics, or engineering, there were a small number from the liberal arts as well. One older friend was completing his dissertation in English — a concordance of the works of George Bernard Shaw. He had a rented keypunch on this front porch because it wouldn't fit through the front door. This was ground breaking stuff in the 1960s.

In the fall of 1964, I began work on my master's degree in physics at the University of Minnesota. Towards the end of my first year, my research professor had me begin doing elastic-scattering calculations in preparation for my thesis research. The process was laborious and required a series of calculations on a mechanical calculator.

The Marchant Calculating Machine Company was founded in 1911 and acquired by the Smith Corona typewriter company in 1958. Within a few years, electronic calculators eliminated the market for such machines, and a few years later personal computers eliminated the typewriter business as well.

SCM Marchant Calculator
Photograph from Wikipedia



A Guide to Fortran Programming
Photograph by Doug Gale

The results of those calculations were then used to look up scattering parameters on tables not too unlike the old artillery range tables or the trigonometric tables in the CRC Handbook. I spent a great deal of time over the first summer completing the calculations that would be needed for my research. When I returned to school in the fall, my advisor said, "Now that you understand how elastic scattering works, let me show you an easier way to do it," and instructed me to buy a copy of Daniel McCracken's "A Guide to Fortran Programming" and to write a computer program to do the calculations that I had previously been doing by hand. That was how you learned to program a computer in those days. I still have my copy of McCracken's brief guide on my bookshelf.

McCracken's *Guide to Fortran Programming*: For two decades McCracken's guides to programming in the Fortran (FORumla TRANslation) programming language were the standard Fortran textbooks and were translated into fourteen languages.

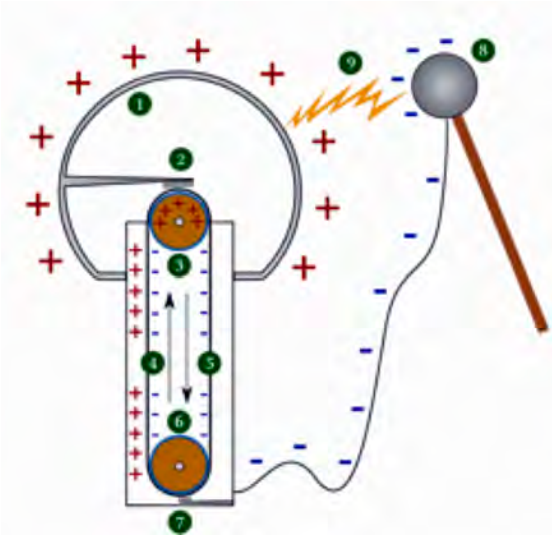
In the fall of 1965, I decided to take a three-quarter course in numerical analysis in the mathematics department to give some formal structure to what I was doing on an ad hoc basis. One assignment in particular made a lasting impression. We were asked to calculate the value of a 4x4 determinant both by hand and by using the University's CDC-6600 mainframe. Control Data Corporation's 6600 was arguably the world's first supercomputer and was designed to solve large scientific problems. Its 60-bit word length in single precision and 120-bits in double precision set the performance standard for many years.

Calculating the determinant exactly by hand was tedious and took the better part of a day, as the 16 elements were all fractions and the lowest common denominator was extremely large. The results of numerical calculations on the 6600, however, ranged from plus infinity to minus infinity! We students were baffled until our instructor demonstrated that the matrix was so ill conditioned that our numerical techniques were useless. I've never forgotten the point he made so dramatically, particularly as I observed the increased reliance on canned numerical analysis programs.

The process of running a computer program consisted of writing the necessary code, usually in Fortran. You would then enter the code onto punched cards on a cardpunch machine. In addition to the code itself, you had to include header and footer cards that contained additional information. These "JCL or Job Control Language" instructions would tell the computer what you wanted done with the program. You would then turn in the punched cards to the computer center; sometime later your deck would reappear, along with the output, on the shelves adjacent to the input window.

It was about this time that I learned the importance of marking the sequence of the punched cards in the deck upon proving the unstated rule, "An unsequenced deck will be dropped and shuffled." Although the last eight columns of the 80 columns of data on a card were reserved for a sequence number, I found it faster to decorate the top of the deck with geometrical drawings done in various colors with felt tipped markers. It also helped in finding my deck on a shelf filled with dozens of other decks.

My thesis work involved nuclear scattering done on a university's Van de Graaff accelerator. My interest in Van de Graaff accelerators began when I was in high school and made one for the Kansas City Science Fair. I naively thought that I could get higher voltages by simply making a bigger machine. I solved the problem of finding a large metal sphere that I could afford by covering a large (3-foot diameter) weather balloon with paper mache painted with conductive carbon paint. The crowned drive pulleys were turned on a homemade lathe, and the belt was driven by a motor liberated from my mother's vacuum cleaner. The whole thing was more than 7-feet tall with the accelerating column made from a large cardboard tube.



Van de Graaff Generator
Illustration from Wikipedia

A metal sphere, atop an insulating column, can be raised to a high voltage by carrying a static charge to the "terminal" on a moving belt. The generator may be used as a particle accelerator by adding a second evacuated insulating column to allow charges to be accelerated from the terminal to the base.

It didn't work very well. The surface of the sphere didn't have a high enough conductance, the accelerating column didn't have a high enough resistance, and there wasn't any mechanism to equalize the potential drop down the column. This was another early lesson that first-order approximations are often inadequate.

The construction of the accelerator at Minnesota begun in 1937 had followed, albeit with far better implementation, the idea that bigger would be better. The accelerator was enclosed in a steel tank filled with compressed air to allow for higher terminal voltages and was three stories tall. The danger was that a spark from the high-voltage terminal would cause a fire inside the tank and raise the air pressure above what the tank could withstand. The external tank remains a fixture on the Minnesota campus. I was the next-to-last student to do their research on the machine, as it was phased out of service shortly after I graduated for a newer accelerator. I found the "hands on" aspect of my research particularly appealing.

University of Minnesota Van de Graaff
Photograph by Doug Gale

The author standing in front of the compressed air shell of the original Minnesota Van de Graaff in 2014, forty-five years after it was decommissioned. It was just too big to move and remains a monument on campus. Each Christmas when it was operational the graduate students would adorn the top of the tank with a brightly lit Christmas tree.

My research made extensive use of a form of computing no longer in fashion --analog computation. Particles were identified by using an electrical analog multiplier to generate a signal proportional to $(E-\partial E)^{0.6}\partial E$, where ∂E was the energy lost by the particle in traversing a gas-filled proportional counter, and E was the energy as it was stopped in a solid-state detector. The magnitude of that signal provided a unique signature for identifying different particles. That information was used in analog circuitry to filter out unwanted interactions. The whole process is now done by recording all "events" and doing a post-analysis on a digital computer.

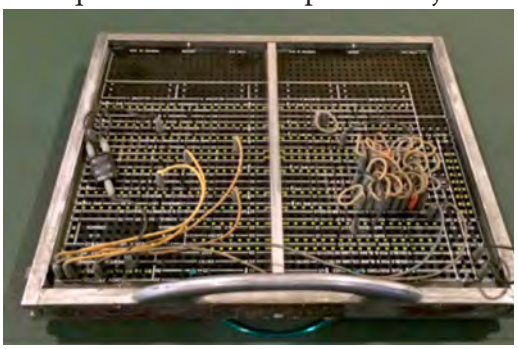
The Late 1960s: Computing Spreads to Smaller Institutions

After completing my master's degree in 1966 I took a break from my education, partially to give my spouse a chance to work on her master's degree and partially to earn some money, and accepted a position as a physics instructor at St. Cloud State College, which was located about an hour from Minneapolis.

While at St. Cloud, I decided that I needed some formal training in computer programming, so when I saw that the University of Minnesota was offering a graduate course in computer programming that was to be offered for three hours one night a week, I immediately signed up. The first class was a bit of a surprise. I was the youngest person in the class. Everyone else was in their 30s or 40s and working for CDC. And the course was CDC machine language programming, not a high-level language such as Fortran. Each week we were given an assignment that was due at class time the next week. Since I was located an hour away and did not have local access to a CDC supercomputer, I would spend the week writing and very carefully reviewing my program. On class day, I would drive to Minneapolis early enough to submit my stack of punch cards in time to get the output before class. The discipline of being meticulously careful not to make mistakes and getting it right the first time were invaluable lessons. Although I didn't know it at the time, the experience of working in machine language would have a profound impact on my later career.

In the mid-60s, using computers in general education courses was relatively uncommon outside a few elite institutions. Programming was considered the specialized province of engineers, scientists, and accountants. I decided, however, to include it in my general education physics classes and prepared a two-page hand out, "Super Simple Fortran." I would cover the material in a single lab period and give each student a personalized assignment due the next day. After much moaning and groaning, the students would always successfully complete the assignment. What I overlooked, unfortunately, was the impact a horde of panicked students would have on the college's limited public keypunch machines.

While I was teaching, my spouse, who had a bachelor's degree in physical education, was taking courses towards her master's degree. Since she had done some keypunching for me the previous year when I was working on my thesis, she decided on a lark to take a programming course. The only one offered was a one-quarter course that covered machine language, assembly language, and Fortran. The instructor was Jack Steingraber, who went on to publish a number of books on computer programming. That course changed her life, as the next quarter she took a part-time job as a computer operator in the college's computer center.



IBM 409 Programming Board *Photograph from Wikipedia*

The college only had two computers, an IBM 409 and an IBM 1620. Both were used almost entirely for administrative computing, although the 1620 saw limited academic use. There was no centralized academic computing support organization. The 409, which was introduced in 1959, was an accounting machine that read punched cards, made simple decisions, and printed the results.

The 409 was "programmed" by moving jumper wires around on a removable panel. To speed changing programs, there were multiple panels, all pre-configured for a particular task.

In contrast to the 409, the university's IBM-1620, which was introduced in 1959 and marketed as an inexpensive "scientific computer," was programmed via a console typewriter or punched cards.

The 1620 was so widely used by scientists and mathematicians that the textbook, *Numerical Methods and Computers* by Shan Kuo, included an appendix of operating instructions for the IBM-1620 that included flipping electrical switches on the console front panel.

IBM 1620 Computer

In 1967 my spouse and I attended a conference in Chicago on the then avant-garde topic of "Computer Assisted Instruction" or CAI. I returned to campus determined to make more use of computers in physics instruction. In one course, I asked the students to write a program to simulate putting a rocket into earth orbit. (*Rocket Trajectory Simulation*, D.S. Gale, Amer. Journal of Physics, 38, 1475, 1970.) The objective was to find, by trial and error, successful launch parameters. While the students learned a great deal, they were unsuccessful in finding a stable orbit. Once again I was reminded of the limitations of numerical approximations. I had one student contact me 30 years later to tell me how that assignment changed his future career to information technology.

My spouse decided to do her master's thesis in education using CAI. Specifically, she wrote a 1620 assembly language program for golf instruction in physical education. The objectives were for players to learn which club to use in different situations and also to learn rules and proper etiquette. The course was then used for a small class of 10 students, who would individually sit at the console responding to queries from the computer. That, of course, required that the university dedicate the entire 1620 to that one application and behind the curtain there was an operator desperately feeding punched cards into the card reader trying to stay ahead of the student. The course was successful and popular with the students.

"Professional educators," however, did not in general embrace computing in the '60s. When my spouse submitted a paper based on her thesis to one of the education journals, it was rejected, because "it is common knowledge that computers are of little importance in education." She was so disgusted that she dropped out of the education community and joined the computing community. This "late adopter" pattern within the K-12 education community would be repeated in the early '80s when microcomputers were introduced.

The Early 1970s: Computing Becomes Ubiquitous for Research at Research Universities

In 1969 I decided to return to graduate school and complete a PhD in Physics. I selected Kansas State University (KSU) because they were installing a new tandem Van de Graaff accelerator, and I would be on the ground floor of building a new machine. One of the attractive aspects of the program was that graduate students ran the accelerator. We were the operators as well as the researchers. If something broke while we were using the machine, it was up to us to fix it.

Data acquisition was done largely with analog electronics much like what I used at Minnesota. The output, however, was punched tape, which we quickly converted to punched cards for further analysis on a digital computer.

KSU Tandem Van de Graaff
Photograph by Doug Gale

A small group of graduate students purchased army surplus cots that we used when we had multiple days of beam time. We lived on pizza and coffee. The tandem was painted purple, one of the school colors.

The late 1960s and early 1970s were a period of political and social unrest in the United States, much of it led by students. In December of 1968 a fire, believed to be set by an arsonist protesting the Vietnam War, had gutted Nichols Hall only a few hundred yards from the KSU physics building and computer center. In April of 1970 the computing facility at the University of Kansas, only 81 miles from Kansas State University, was bombed. (http://kuhistory.com/articles/this-is-no-joke/) The explosion and subsequent fire injured three students and caused extensive damage. Several months later, a science building at the University of Wisconsin was bombed and resulted in the death of a physics researcher. (http://en.wikipedia.org/wiki/Sterling_Hall_bombing)

Since Kansas State's accelerator was located beneath the computer center, the graduate students working on the accelerator were concerned for their safety. The faculty felt that the radiation warning sign and locked door leading down to the accelerator were sufficient protection. We graduate students were less sanguine and prepared an evacuation route via a metal ladder that led to a hatch above the accelerator vault. We also had a stash of iron pipes for personal defense, if needed.

The theoretical portion of my dissertation required running two very different computer programs. The first was to determine the optical model parameters that best described the scattering of the colliding particles to first approximation. While the calculation was straightforward, it had to be done many times because the process basically consisted of guessing two variables, calculating the results, and then comparing the results with the experimental data. The difficulty was that there were multiple variable pairs that provided an acceptable fit. The challenge was to find the best pair. The process was much like finding the deepest valley in hilly terrain on foot. Running the program during the day on the university's IBM 360-50 was not an option because of the associated computing charges. However, the computer center made time available at night when the machine was idle. For over six months, I always had several programs waiting to be run overnight. It helped that my office was across the hall from the computing center, and my wife was manager of user services.

The IBM 360-50 was widely used at colleges and universities during the late '60s and '70s. The IBM System 360 "family" of computers used a consistent architecture for computers with different price points, and was very successful as it allowed customers to purchase a smaller system and migrate to a larger one as needed. The architect of the 360 family, Gene Amdahl, later founded the Amdahl Corporation, which directly competed with IBM. The development of the System 360 was a "bet the company" decision, as the development costs were twice IBM's annual revenue at the time.

The project manager, Fred Brooks, became famous for his book, *The Mythical Man-Month: Essays on Software Engineering*, that became required reading for new technology managers. The book is most remembered for Brook's Law, which states "adding manpower to a late software project makes it later."

The 360-50 could perform up to 48 kiloflops (floating point operations per second). This contrasts to Apple's 2015 MacPro, which peaked at 6 teraflops—more than a hundred-million times faster.

IBM 360-50
IBM Archives

The second program required in my dissertation research was much more computationally intensive. It involved solving coupled partial differential equations. The program could not be run on the university's IBM mainframe, even if they had been willing to give me the required time. The 360 had a 32-bit word length and researchers had found from hard experience that 48 bits were the absolute minimum to solve the equations numerically. (One researcher spent the better part of two years unsuccessfully trying to get the program to run on the System 360 architecture.) Since the program was running successfully on a CDC 3600 computer at Argonne National Laboratory, the decision was made to send me there to complete the calculations. When I left I was given $30,000 in computing funds—which I burned through quickly—and was given another $20,000. That too went quickly, and I requested the University send more money. They did, $10,000, but said that if that didn't get results to not come back. Fortunately, I was able to reach closure. The whole experience illustrates how researchers at the time were challenged to do the necessary calculations on the available computers.

Shortly before I graduated, the nuclear laboratory got a mini-computer, a Digital Equipment Corporation or "DEC" PDP-11, to replace its obsolete paper-tape system and much of the analog electronics. I made a deliberate decision not to get involved with the new computer and graduated on schedule.

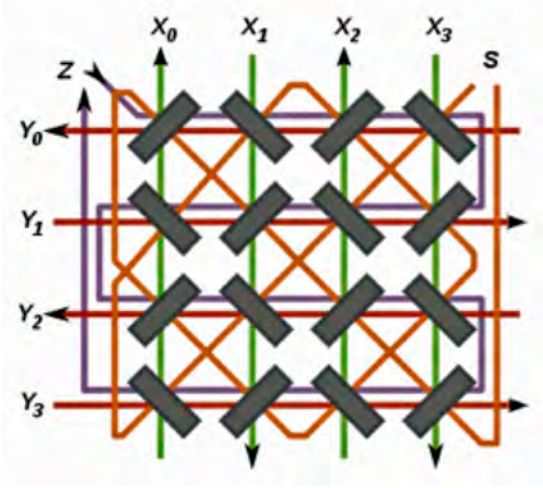
The Mid 1970s: Computing Becomes Ubiquitous at Smaller Institutions

In 1972 I accepted a position as Assistant Professor of Physics at East Texas State University (now Texas A&M-Commerce). While there I quickly became the "user from hell," consuming around 30% of the cycles on the University's IBM 360-50 mainframe, as I continued to analyze data I had compiled at Kansas State. While the university had a computer center, its primary focus was supporting administrative computing. Academic users were pretty much on their own. The center had in place a structure for supporting research computing without real dollar charges for times when the computer was not running administrative jobs. Other than myself, there was little research computing being done on the University mainframe.

The output from my calculations on the university's 360-50 were sent to a CDC computer somewhere in Maryland via satellite link, where more numerically intensive, coupled-channel calculations were done. I don't recall what the funding mechanism was to pay for those cycles. As a researcher, I didn't care where the computer was located or how it was paid for; all I was interested in was the results. This cavalier attitude later helped me understand researchers' attitudes when I became a computer center manager.

I spent the summers of 1973 and 1974 at Argonne National Laboratory as part of a visiting research program. They had a tandem Van de Graaff and my research was based on bombarding various exotic isotopes of uranium with lithium ions. Some of the uranium isotopes were quite rare in nature, and required "cooking" more common isotopes in a nuclear reactor for as long as two years.

The Argonne accelerator had a home-built computer system for a much more sophisticated form of data collection than I had used earlier. Rather than using analog circuitry to filter out signals from interactions that were not being studied, all signals for all reactions were recorded on the computer and the unwanted reactions filtered out in post-analysis on the computer. This "event recording" strategy meant that the stored data could be reanalyzed for other studies.



The primary memory was built from a surplus magnetic-core unit that was physically bigger than I was. Magnetic-core memory was the predominant form of memory at the time and consisted of a matrix of very small toroids (doughnuts), called cores, which could be magnetized either clockwise or counterclockwise by an electric current through wires in the center of the cores.

Core Memory Wiring Scheme
Illustration from Wikipedia

This diagram shows a simple scheme for magnetizing and subsequently reading the magnetic state of individual cores. Even though magnetic cores are no longer in common use, we still use the term "core memory."

After the event data was analyzed for the particular reaction being studied, the information was transferred to magnetic tape for further analysis.

The summer stays at Argonne were augmented by trips to Argonne during the school year. Typically, I would travel with a suitcase of magnetic tapes and the clothes on my back. It was during one of my stays at Argonne that I purchased my first pocket calculator for $100 (a bargain price at the time) at Marshall Field's Department Store. It could add, subtract, multiply, divide, and calculate square roots.

After the two summer appointments at Argonne, I continued to get beam time on their accelerator. Utilizing that time, however, was exhausting. Along with one or two students, I would drive straight through from Texas to Chicago, and then spend the next 24 to 48 hours at the accelerator. That would be followed by spending another 24 to 48 hours analyzing the event data and reducing it to a format that could be transferred to magnetic tape. Unfortunately, that could only be done on the home-brew computer at the lab. Thus, after three to five sleepless days, we would drive straight through to Texas and try to catch up on missed classes.

Upon return to campus, I would transfer data from magnetic tape to punched cards for easier use in running analysis programs on the local mainframe. As I recall, at one point I had over a million cards neatly stored in filing cabinets. Unfortunately, East Texas is humid and I had a serious problem with cards absorbing moisture and jamming in the card reader. At one point I had tables set up outside in the sun with thousands of cards drying out.

The grueling routine ultimately resulted in changing my research focus to something that could be done locally, and ultimately led to a career switch. If the Internet had existed, allowing data to be easily transferred and the experiments to be run remotely—and local campuses had access to high-performance computing resources remotely—I would probably have finished out my career as a nuclear physicist.

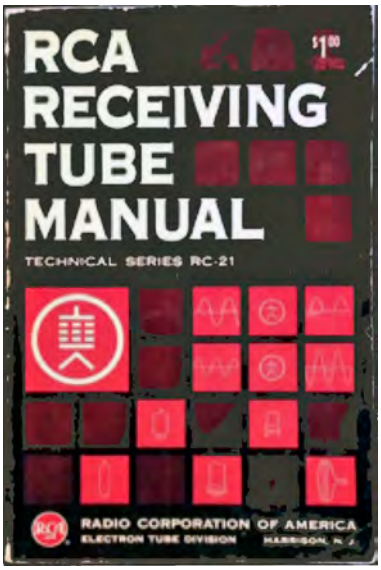
In the 1970s there were six U.S. mainframe computer companies: IBM, which had most of the market, and five others called the BUNCH, which was the nickname given to Burroughs, Univac, NCR (National Cash Register), CDC (Control Data Corporation), and Honeywell. In the 1960s, RCA and General Electric also made computers, and the group was characterized as "IBM and the Seven Dwarfs." Increasingly through this period, IBM dominated the market.

The early 1970s also saw the emergence of the minicomputer as a computing alternative for small companies or research groups. An early definition of a minicomputer was a machine whose cost was in the five-digit range. They had their own architectures and operating systems, and were commonly used for device-control and instrumentation. Digital Equipment Corporation, or DEC, was particularly prominent in higher education and research. The term "VAXinated" became a common joke in the research community.

The physics department at ETSU took advantage of my somewhat eclectic background and assigned me to develop an observationally based, introductory astronomy course, and a musical acoustics course targeting music majors. The department's budget was tied to credit-hour enrollment and such courses were essential to maintaining a viable and rigorous physics program. Both courses were very popular, even though I had a reputation as a tough grader.

I was also assigned to teach the undergraduate "electronics for scientists" course that was routinely taken by undergraduate physics majors. The focus of the course was instrumentation of experimental science using transistors and integrated circuits. When I began teaching the course, there was a shortage of textbooks on the subject, so I taught largely from my own notes.

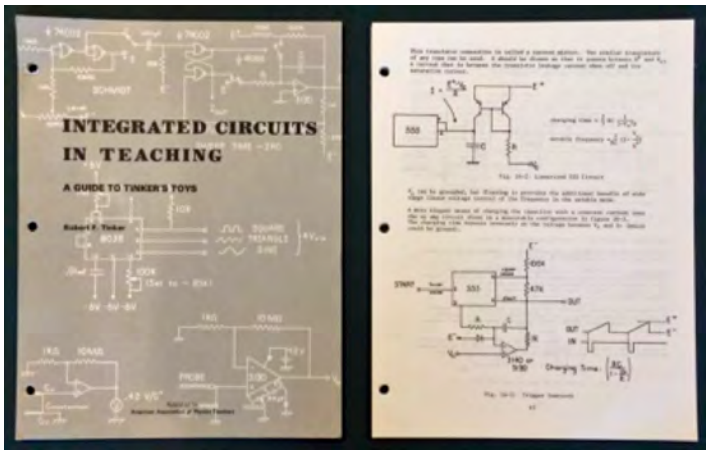
When I took the corresponding course as an undergraduate a decade earlier, the applied portion focused almost entirely on vacuum tube electronics, even though the transition from vacuum tubes to transistors was well under way. By the time I taught the course a decade later, vacuum tubes had almost entirely been replaced by transistors and integrated circuits.



RCA Vacuum Tube Manual
Photograph by Doug Gale

During this period I attended a "Chautauqua" Workshop developed by Robert (Bob) Tinker, of the Technical Education Research Center and the American Association for the Advancement of Science, and funded by the National Science Foundation. The workshop targeted scientists and engineers, and focused on using integrated circuits and digital logic for device control and data recording. Tinker and I shared a return airplane flight and spent the time talking about hardware modifications needed to convert an analog color TV to a digital color monitor -- something that was a rarity at the time.

Tinker's Chautauqua workshop was taught from mimeographed loose-leaf handouts and reflected the lack of commercial textbooks.



Tinker's Toys
Photograph by Doug Gale

By 1974 I had begun introducing simple digital electronics into the undergraduate "electronics for scientists" course, and shortly afterwards began teaching a one-semester "digital electronics" course organized around the newly released Hewlett-Packard Model 5035T Logic Lab. The course was popular with computer science students as well as science students.

HP had the challenge (and opportunity) of teaching an army of analog-trained electrical engineers about digital electronics. They developed a training course complete with lab hardware and instruction manuals. Students would construct various digital circuits on a plug-in board. The lab included an assortment of digital measurement and probe tools. The objective, of course, was to sell HP test gear.

HP Model 5035T Logic Lab
Hewlett-Packard Archives

The decade also marked the emergence of the microprocessor, in which an entire central processing unit (CPU) was implemented on a single chip. That, combined with the increasingly less expensive 7400 series logic chips led to the development of the microcomputer, a term used in the 1950s by the science fiction writer Isaac Asimov. The marketplace was crowded with companies developing new microprocessors and support chips. The Zilog Z-80, MOS 6502 Motorola 6800, and Intel 8080 chips in particular led to the development of a wide range of consumer computers, including the Altair 8800, Radio Shack TRS-80, and Apple I.

Hewlett-Packard wasn't the only company to market training materials for their digital technology. Other chip vendors introduced educational "training kits."

Intel sold a number of heavily subsidized "kits" to encourage users to adopt their new products. This picture shows their bubble memory prototype kit, which showcased their 1-megabit bubble memory module. The kit included a printed circuit board and a bag of parts.

Bubble Memory Prototype Kit
Photograph by Doug Gale

Intel provided a System Development Kit (SDK) when they launched a new microprocessor. It was a natural follow up to the digital logic course to begin teaching a microcomputer-based course on the SDK-85 in 1976.

The SDK-85 was a single board microcomputer system using the Intel 8085 chip, which was based on the popular 8080. It came as a kit and the completed circuit board included a 6-digit LED display, a simple IO bus, a 24-key keyboard for direct insertion, examination, and execution of a user's program, and a large space for wire wrap components.

SDK-85 1Single Board Computer
Photograph from Wikipedia

The course was organized as a "hands on" graduate

workshop that met for 3 hours once a week in the evening, to allow local professional engineers to attend. The students would assemble the single board computer the first evening and were given assignments of increasing complexity as the course progressed. The computer could be programmed directly from the keypad or through a serial port. As the students progressed, the programs became more sophisticated and included attaching various input and output devices.

When I first offered the course, there weren't any suitable textbooks so teaching was done from manufacturers' technical literature and handwritten lecture notes, such as the ones shown here. Typical labs included the construction of a digital thermometer and the control of a model radar tower made from Erector Set parts borrowed from my son. More sophisticated labs employed analog-to-digital and digital-to-analog conversion circuits. By the end of the decade textbooks began to become available, but having written a Laboratory Manual while at St. Cloud, I had no desire to undertake converting my notes to a textbook.

SDK-85 I/O Notes

Teaching Materials
Photograph by Doug Gale

The course proved to be popular not only with physics and engineering students, but with computer science students as well. So much so that the course credit could be taken either in physics or computer science, and led to a joint-degree program offered by the two departments with courses in microelectronics and microcomputers. A student could major either in physics or computer science and minor in the other discipline. (*Integrating Microcomputers and Microelectronics into the Physics Curriculum*, D.S. Gale, Amer. Journal of Physics 48, 498, 1980.) One thing led to another, and I began teaching a graduate Telecommunications course in the Computer Science Department —something that would prove useful later in my career.

"Amateurs" and "hobbyists" played a vital role in microcomputing during the mid- and late 1970s. Some were young entrepreneurs like Steve Jobs and Bill Gates, who went on to found major corporations. Others were working engineers and scientists who wanted to exploit the opportunities presented by the new technology. This period could be viewed as a precursor to the early 21st-century "maker movement," which enjoyed engineering–oriented pursuits such as electronics, robotics, and 3-D printing, in addition to more traditional arts and crafts. Many of today's computer science luminaries gained their experience at places like Bell Labs and the Rand Corporation rather than at academic institutions.

The microcomputing culture had its own "do-it-yourself " magazines and paperback books.

Byte Magazine
Photograph by Doug Gale

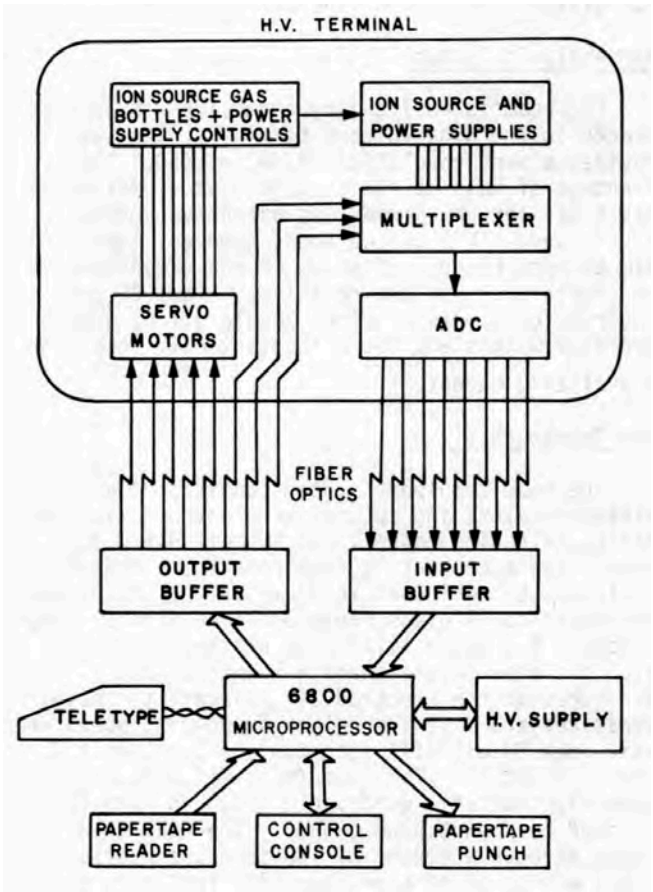
Byte magazine was a monthly publication dedicated to microcomputers. It was widely read by both computer hobbyists and professionals, and its articles covered both hardware (complete with wiring schematics) and

software (usually machine or assembly language). *Interface Age* was another popular magazine that targeted the "home computerist."

This period also marked the publication of a wide range of "cookbooks" that stressed practical electrical circuits to supplement the large amount of similar "how to" manuals being provided by the chip vendors. Again, these "cookbooks" were an eerie precursor to the cut-and-paste approach favored by the 2010's maker culture.

Cookbooks
Photograph by Doug Gale

Although having fun playing with digital electronics, I had not abandoned physics research. In 1975 I began the design and construction of a 300-kilovolt heavy ion accelerator. The scientific objective was to study atomic reactions on the surface of material by bombarding them with heavy ions; the personal objective was to move my research closer to home. The design of the accelerator, particularly the ion optics of the accelerating column, involved extensive numerical modeling and was the focus of one of my graduate student's research.



The accelerator had several unique features at the time. The first was the use of then state-of-the-art, ultra-high vacuum techniques that totally avoided the use of any organic materials. The second was the use of fiber optics to transmit signals to and from the high-voltage terminal. The fiber optic connectors that became common in the 1990s were not available. Fiber optic strands were hand polished and then epoxied onto LED transmitters and detectors. The third was the extensive use of digital electronics and microcomputers for telemetry and voltage control.

300 kV Heavy Ion Accelerator Telemetry
Scientific & Industrial Applications of Small Accelerators, 4th Conference, 1976

When I ran out of grant money to purchase a needed piece of data acquisition electronics for the 300 kV heavy ion accelerator that was halfway to completion, one of my students, who was also a teaching assistant in the microcomputer course, suggested that we build a computer system and I/O electronics that would replicate the functions of the dedicated electronics we could not afford. In an act that can only be viewed as hubris, I agreed.

We used the astronomy dark room to make printed circuit boards. We added stepper motors to the milling machine in the physics shop to drill holes in the boards. We cannibalized military-surplus disk drives for memory. We knew enough to be dangerous -- and very creative. At the time, I jokingly said I didn't know what an operating system was until I wrote one.

In the summer of 1979 I was offered the position of Director of Decentralized Academic Computing Services at Cornell University. Because of teaching commitments, I was not able to move to Ithaca until December of that year. Although I had illusions at the time of continuing my physics research, in fact it marked the end of my career as a research scientist and the beginning of my career as an information technologist and research administrator.

Chapter 2:

The Early '80s and the Microcomputer Revolution

By the 1980s computing was well established in higher education. Leading institutions were spending approximately 5% of their budget, exclusive of ancillary operations such as dormitories and athletics, on computing. Expenditures were more or less evenly split between academic and administrative applications. Administrative computing was done almost entirely at a central computing facility; most academic computing was done there as well. Faculty and students were still learning about the technology, but applying it in new and novel ways. However, change was on the horizon.

The Cornell Years: Distributed Academic Computing Services (DACS)

The early 1980s marked the beginning of a fundamental shift in computing. The introduction of the minicomputer a few years earlier made it possible for a small department to own its own computer, separate from the centrally managed corporate mainframe. Similarly, the introduction of microcomputers made it possible for individuals to own a personal computer.

The fundamental paradigm shift was not one of hardware but of management control: from centralized to decentralized. The term "microcomputer" fell out of favor in the mid-80s and was replaced with "personal computer or PC" reflecting that the fundamental shift was not based on computing power but on management control.

Cornell was one of the first universities to recognize that a paradigm shift was underway and decided in early 1979, under the leadership of Douglas VanHouweling, Director of Academic Computing, and J. Robert Cooke, Chair of the Faculty Computing Committee, to recommend the creation of a separate division within Cornell's computer services unit to support decentralized computing. Provost Kennedy issued a policy directive in July of that year creating DACS, for Distributed Academic Computing Services, within the Academic Computing unit. The unit was charged with providing consultation and support for those units owning or planning acquisition of computers.

The first director of the DACS was Alison Brown, who quickly decided she would rather not be burdened with administrative responsibilities, and recruited me in the fall of 1979 to be her replacement and boss. My wife, kids, and I arrived in Ithaca two days before Christmas in 1979.

When Ken King became Vice Provost for Computing at Cornell in late 1980, DACS was moved from the academic computing unit to directly reporting to the Vice Provost, and the name was changed to Decentralized Computing Services, or DCS, because administrative users were following the same trends towards decentralization. King cites Cornell's recognition of this paradigm shift as one of the reasons he accepted the position at Cornell. At the time, computing centers typically had four branches: academic computing, administrative computing, systems, and operations, and had a centralized management culture.

A Rude Awakening

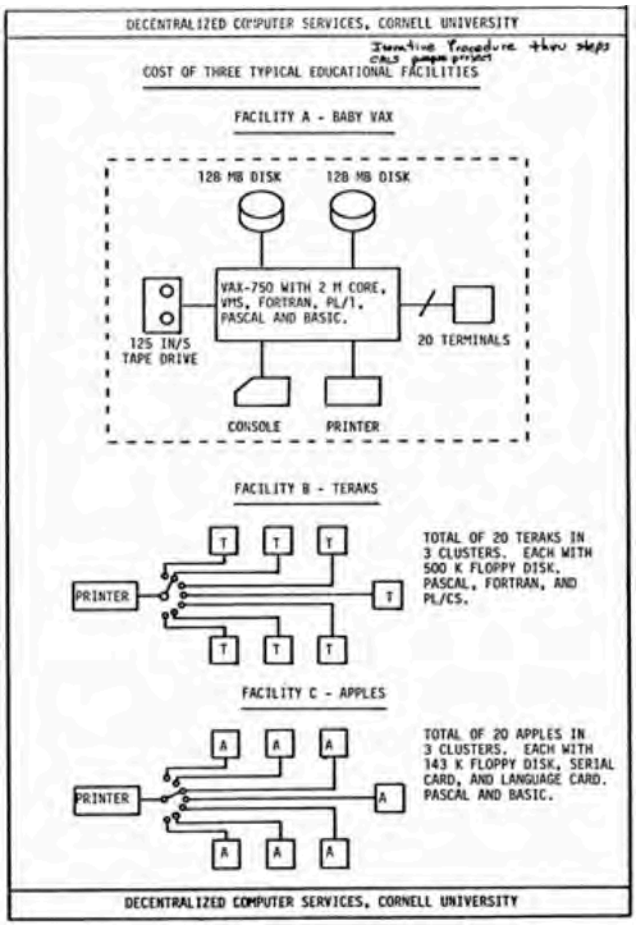
Shortly after arriving at Cornell I was asked to participate in the selection of a new minicomputer for the Business School. They were considering two architectures, both from Digital Equipment Corporation, a DECSYSTEM-20 and a VAX. I quickly realized that, while I knew a lot about microcomputers, I was a real neophyte when it came to minicomputers. The decision boiled down to choosing between an established architecture

that had a large body of mature software, and a newer architecture with relatively little application software —the kind of decision that required experience I didn't have.

My response was to attend a two-day workshop on minicomputers and microcomputers in Washington, DC, to spend a lot of time talking to facility managers, operations staff, and minicomputer owners, and then to develop a one- to two-day workshop syllabus of my own: *Guidelines for the Selection and Operation of Minicomputers and Microcomputers.* I subsequently gave presentations and workshops of various lengths targeting different audiences at a variety of venues, both local to Cornell and nationally, for meetings such as the National Educational Computing Conference and the American Association of Physics Teachers, using a subset of the complete syllabus. The syllabus had seven sections:

1. Introduction: A Brief History of Small Computers and an Explanation of Some Common Elements in "Computer Talk"
2. The Relative Advantages and Disadvantages of Micro/Mini/Midi/ and Maxicomputers
3. How to Determine What Kind of Computer is Best for You
4. A Brief Survey of Hardware and Software
5. How to Select and Acquire a Small Computer System
6. Warranty, Maintenance, and Operating Costs
7. Future Trends

It is important to remember that most of the attendees at these presentations had little experience with computing, and for those that did, it was usually restricted to using a mainframe or a home computer. Few understood the tradeoffs between mainframes, minicomputers, and microcomputers, and virtually none had an understanding of the total cost of ownership or maintenance and support costs and issues.



The first two sections covered basic definitions and the relative advantages and disadvantages of Micro/Mini/Midi/ and Maxi computers. Topics included cost, hardware and software capability, and the number of users that could be supported.

The objective of the next three sections was to assist the attendee in selecting the appropriate kind of computer. Included was a brief survey of then-current hardware and software, costs for hardware and software, as well as how to prepare and evaluate an RFP or RFQ. Benchmarks for various CPU processors and operating systems were presented in detail.

The fifth section was an attempt to introduce to attendees a sense of how much it really cost to operate a computer and/or a computer facility. This section included comparisons of the costs of running typical student facilities, such as a baby VAX (VAX-750), a cluster of 20 Teraks, and a cluster of 20 Apple IIs. I spent a considerable amount of time in this section, because I had quickly discovered that many faculty attendees thought they could avoid "excessive" mainframe charges by purchasing a bunch of microcomputers from the local computer store, without giving thought to things like software, maintenance, power, cool-

ing, site preparation, and printers. And they rarely included funding for staff support. I'm not sure how successful I was in convincing attendees that there was no such thing as a free lunch.

Benchmarks

But users wanted more than theoretical background; they wanted concrete advice, such as "buy product A." In the early 1980s the microcomputer market was in a state of flux. There was a wide array of CPU chips from multiple vendors. There were few, if any, operating system or software standards. In DACS we spent a great deal of time benchmarking various systems in an attempt to provide our users with the best advice possible. This is a benchmark summary document I prepared in the fall of 1980; the subjective opinions were mine. For obvious reasons, it was always presented "in-person."

A Comparison of Some Popular Microcomputers. D.S. Gale, 10/80. The numerical "ratings" are the "seat-of-the-pants" opinion of the author.

Ratings Scale: 1=Terrible 3=Average 5=Excellent 2=Poor 4=Good

Computer	CPU	Word Size	RAM	Base List Price	Typ. List Price	Princ. Prog. Lang.	# inst. to Spr '80	Primary Uses	Operating System	Evolutionary Future	"Power"	Expandability	Local Software Support	Natl. Software Support	Local Hardware Support	Natl. Hardware Support	Reliability
Apple II and Apple II Plus	6502	8 bit	16K-48K	1.2-1.5K	$2K 32K minidisk CRT	BASIC PASCAL Assemb.	55K	Ed., bus, home/hobby	DOS	2 (CPU)	3	3	3	3	3	3 I/O periph.	3
Commodore PET	6502	8 bit	8-32K	0.8-1.3K	$1K 8K cassette CRT	BASIC Assemb.	107K	Ed., home/hobby	in ROM internal	2 (CPU)	2	2	1	2	1	2	2
Cromemco Z-2	Z-80A	8-bit	4-64K	1.3-2.2K	$4K 16K minidisk CRT	BASIC FORTRAN COBOL Assemb.	NA	Small Bus, Control, Prof.	CDOS (CP/M compat.)	3 (CPU)	3	4	3	3	3	3 I/O periph.	4
Heath H9	6080A	8 bit	8-56K	0.5-1.6K	$1.6K 16K minidisk CRT	BASIC Assemb.	12K	home/hobby, ed, bus	cassette OS	2 (CPU)	2	2	1	2	1	2	2
Hewlett-Packard HP-85	HP nMOS	8 bit	16-32K	3.2K	$3.2K 16K RAM 2.8K tape Printer CRT	BASIC	NA	Sci./Eng. Prof.	HP-85 OS internal	4 (package)	3	4	2	3	2	5 I/O periph Service	5
TRS-80 Model I	Z-80	8 bit	4-48K	0.5-1.5K	$1.5K 16K minidisk CRT	BASIC FORTRAN Assemb.	150K	home/hobby, ed, bus	TRS-DOS	2 (sound package)	3	3	2	3	1	5 Service	2
TRS-80 Model 2	Z-80A	8 bit								3 (CPU)	3	4	2	3	1	5 Service	3
Texas Inst. TI 99/4	9900	16 bit	16K	1.1K	$2.1K 16K minidisk	BASIC Assemb	NA	home/hobby, ed, bus	internal in ROM	2 (package)	3	1	1	2	1	3 Service	2
Terak 8510	LSI-11	16 bit		$6K	$6K 56K RAM 1MB disk CRT	PASCAL FORTRAN RT-11 APL, BASIC	NA	Sci./Eng. Ed., Prof.	PASCAL, RT-11	4 (CPU, software)	4	5	4	4	3	5 I/O periph Service	4

The list includes neither the Apple MacIntosh nor the IBM-PC, as they had yet to be introduced.

Personal Computers (PCs)

Complete microcomputer systems began being mass marketed in 1977 with the advent of the Apple II, Radio Shack's TRS-80, and the Commodore PET. These "home computers" usually ran the BASIC computer language and used a cassette tape recorder for external memory. Although the Apple II was popular in the educational community, most universities and almost all businesses turned up their respective noses at these machines. They weren't "serious computers." Two events changed this mindset. The first was the release of VisiCalc, the first spreadsheet computer program for microcomputers, in 1979. Overnight, it changed the Apple II from a hobbyist "toy" to a serious business tool. I recall wanting an accounting system to parallel the university's official accounting system. I was seeking a way to project anticipated events into a long-range

financial planning model. I was told the system would take 6 months to develop on the university's mainframe database. Instead, I checked out an Apple II with a copy of VisiCalc. That evening I learned VisiCalc and wrote the planning model.

The second event was IBM's introduction of the IBM Personal Computer, or PC, in August of 1991. The introduction gave microcomputers instantaneous credibility. In fact, the term "microcomputer" dropped from users vocabulary almost overnight and was replaced by the term "personal computer." Big Blue had spoken.

One of the most enjoyable aspects of the job was the opportunity to play with all the new stuff.

Introduced in 1981, the Osborne I was one of the first "portable" computers. The front hinged down to reveal a 5-inch screen, keyboard and dual, single-sided floppy disk drives. I can still recall walking across campus with this 23-pound brute.

Osborne I
Photograph from Wikipedia

Nibbles

Initially DACS was quite small: myself, Alison Brown (a systems' guru), Karen Friedman (a software guru), and Joe Nazar (a hardware technician). The challenge was dealing with an overwhelming number of faculty and staff interested in using microcomputers, but without any experience in personal computing. At one point, in desperation, we resorted to waiting until noon to open the office because of the numbers of faculty and staff waiting to talk to someone in person. To leverage our limited resources, we created a one-page folksy newsletter called "nibbles," which covered the issues that people were concerned about at that time.

dacs

nibbles

Decentralized Academic Computer Support — Ithaca, New York

introducing dacs...

...and nibbles

Circle October 15

A Publication of CORNELL COMPUTER SERVICES. September 1980 Volume: 1 Number: 1

The Great Debate

Shortly after I arrived at Cornell I was asked to participate in a traveling debate sponsored by the business-oriented Data Processing Management Association (DPMA), which changed its name to the Association of Information Technology Professionals in the 1990s. The debate was between decentralized computing, using minicomputers and microcomputers, and centralized computing using mainframes. I would argue for a decentralized computing environment, while a member of the hosting DPMA chapter would champion centralized computing. The discussions were "lively" and underscored that the fundamental shift was centralized versus decentralized and not mainframe versus minicomputer or microcomputer.

Word Processing

Word-processing in the late 1970s and early 1980s was done on dedicated and expensive stand-alone machines, such as the Wang 1200 and Xerox 860, because the hardware and software capabilities of available microcomputers, such as the Apple II, were very limited. My wife, who was part of the administrative computing unit at Cornell, provided Xerox 860 technical support to the College of Arts and Sciences, before moving on to implementing a student information system.

The Xerox 860 was a high-end word processing system that cost around $14,000 and was popular in the early 1980s. It featured "WSIWYG" or "What You See Is What You Get" software and could display 70 lines of 102 characters on the monitor.

Xerox 860 Word Processor
Photograph from Wikipedia

By the mid-1980s there were dozens of WSIWYG word processing packages, such as "WordStar" that ran on personal computers like the IBM-PC, and the market for stand-alone word processors collapsed.

The Apple Lisa

The Apple Lisa is little remembered but played an important role in the evolution of microcomputers. Steve Jobs had visited Xerox's Palo Alto Research Center in 1979 and was very impressed with the mouse-driven graphical user interface (GUI) that they had developed. When he returned to Apple, they began development of a Motorola 68000-based computer to transform PARC's research into a marketable product. The result, the Lisa, was introduced in January of 1983. The unique feature of Lisa was the integrated package of software applications: LisaWrite, LisaCalc, LisaDraw, LisaGraph, LisaProject, LisaList, and LisaTerminal—and an integrated package that was not available elsewhere. A Lisa was on my desktop at Cornell.

The Apple Lisa
Photograph from Wikipedia

Priced at $10,000, the Lisa was considered by the market place as "too expensive." Ironically, an equivalently equipped IBM-PC XT (introduced in March of 1983) was also about $10,000, and the software applications were not integrated. Relatively few Lisa computers were sold, although at Cornell they were adopted by the Cornell School of Hotel Administration as part of a pioneering system for hotel administration.

Microcomputers in Instruction: The Terak Story

Cornell was one of the first universities to embrace microcomputers in undergraduate instruction. The Computer Science faculty were strong advocates of structured programming and foreswore languages such as Basic,

that were used in most microcomputers at the time. Tim Teitelbaum, in the Computer Science Department, had developed a PL-1 synthesizer that ran on the university's IBM 360-168 mainframe and was used in a course required of all engineering and most science freshman. The synthesizer created a "padded cell" environment that simply did not allow the student to write unstructured code.

See: An Oral History Conversation: The Paradigm Shift from Centralized to Decentralized Computing at Cornell. King, Kenneth M.; Cooke, J. Robert; Teitelbaum, Tim; Gale, Doug. http://hdl.handle.net/1813/41195

In the late 1970s, to alleviate strain on the university's mainframe facilities, Tim developed a version that could run on the Terak 8510, a microcomputer. The Terak was considerably more powerful than contemporary microcomputers based on Intel, Zilog, and Motorola CPUs. One downside was that they cost around $5500, as I recall. Another was the 40+ pound weight of the base-processing unit alone.

Cornell made a major investment in Terak computer rooms to service the introductory programming course that Tim taught and was required of all engineering students. DACS was charged with running the facilities. The problem was that there were never enough Teraks. The days before an assignment was due were chaos. The undergraduates hired to run the facilities and ration use were faced with students concerned about passing a required course. We could measure how crowded the facilities were by counting the number of fights that broke out as students vied for machines. My efforts to get the various Deans to commit funds to expand the facilities were generally unsuccessful. It was always someone else's students who were using the facilities.

Terak 8510/a
Photograph from Wikipedia

This was a new variant of the old problem of how do you pay for computing. The new twist was that we didn't have an accounting mechanism that would allow us to use the somewhat-effective mechanisms that we developed in the '60s and '70s. Other than paper sign-in sheets maintained by the computer room assistants, we didn't know who was using a Terak. And once the student was on a machine, we didn't have a clue as to what he or she was doing. We did try to manually limit each student to an hour of machine time. Before committing additional funding, the Deans wanted evidence that it was their students who were using the machines and how much they were using them.

The Teraks were built with a DEC LSI-11 processor and could run a basic version of UNIX, as well as UCSD Pascal. It had a graphic's processor and could display both text and graphics in monochrome.

The Apple Logon Machine Fiasco

Our attempts to use the older paradigm of "charging by the drink," by adding an accounting function to microcomputers that duplicated time-sharing's logon procedure, was a complete flop. Next to each Terak in our labs we placed a small box with three lights: red, yellow, and green. Each box was hardwired to its associated Terak's interrupt line, and to a central Apple II microcomputer that was similarly connected to each Terak in the room. We developed software for the Apple, using an obscure structured language developed in Europe, to provide authentication, data recording, and most importantly rationing functions. When a student wanted to use a Terak, they would logon to the Apple II logon machine by entering their name and password. Since the memory capacity of the Apple II was limited, the password was not verified against a database. Rather the

student's password was an encrypted version of the student's name. The encryption algorithm was our secret. When a machine was available, the student's name was flashed on a monitor, the associated Terak was enabled, a 60-minute timer was started, and the box beside the Terak showed green. When a student had 5 minutes left the light turned yellow, and when his or her time was up, the light turned red and the machine disabled. Brutal, but effective.

It actually worked—as long as the load on the system was light to moderate. For reasons we could never discover, the Apple software we had developed would crash when the load was high—typically the evening before an assignment was due. We succeeded in making the problem worse.

The solution to the problem was the gradual migration away from a "pay by the drink" funding model, the adoption of less expensive Apple MacIntoshes for the introductory computer science course in 1984, and the widespread ownership of personal computers.

Maintenance

In the 1980s computer maintenance was a major expense. The yearly maintenance cost for a mainframe typically was in the six digits. Maintenance was also a major expense for microcomputers, particularly in the early 1980s. Because of the mild summers in Ithaca, many of the buildings at Cornell were not air-conditioned, and temperature was simply regulated by opening or closing windows. Unfortunately, a Terak and a person each put out around 150 watts of heat, so a 30-student lab generated approximately 9,000 watts of heat. One lab in the engineering building was particularly troublesome. It only had a few windows and on a calm day it would become uncomfortably warm, and the Teraks would start failing as their integrated circuit chips overheated. Tom Everhardt, the Dean of Engineering and all around fine fellow, and I went 'round and 'round on the need to air-condition the room. I went so far as installing thermistors on the surface of the CPU chips to monitor and document the fact that chips were failing because of the heat. In 1980, a room filled with microcomputers and students was something of a novelty.

At $10,000 a pop, the Teraks were not a consumer item that you could buy at the local computer store. The company was small and did not have a network of national service centers. To repair one, you had to box it up and send it back to the factory in Arizona. (Remember, the base unit weighed more than 40 pounds.) That was expensive and had a turnaround time measured in weeks. We quickly decided that it would be cheaper to send one of the DACS technicians to Arizona to work at the factory for several weeks and learn how to repair them, and then maintain a local inventory of parts. Upon the technician's return from Arizona, we borrowed a VHS video recorder and on a Saturday afternoon, fueled with takeout sandwiches and a six pack of beer, put together a cheesy "How to Fix a Terak" video. Although it was originally intended for our own internal use, as other universities learned of its existence it became widely requested.

Email, The Killer App

Although email was being widely used by the small ARPAnet (Advanced Research Projects Agency Network) community, and within a few companies such as IBM, it was just emerging in the academic community in the early 1980s. There were two technical challenges. First, how do you exchange information between two computers, and second, how do you present that information to an end user on a computer terminal.

Cornell's primary computing system in the early 1980s was an IBM 360/168 running the "Virtual Machine" or VM operating system, which allowed the machine to run multiple guest operating systems, such as CMS (Conversational Monitor System) or MVS (Multiple Virtual Storage), each within its own virtual machine. One such system was RSCS (Remote Spooling Communications Subsystem) that was designed to exchange files between remote systems and users.

Using RSCS as the interconnecting mechanism, Ira Fuchs at the City University of New York and Greydon Freeman at Yale University created BITNET, which originally stood for "Because Its There," in 1981. (http://en.wikipedia.org/wiki/BITNET) BITNET was a point-to-point "store and forward" network. Files were temporarily stored on one computer and then transmitted to another computer at a later time over a leased telephone line. A handful of other universities, including Cornell and the University of Pennsylvania, quickly joined the new network.

The business model was simplicity itself. If a university or college wanted to join BITNET, they paid for a 9.6 kbps (thousand bits per second) leased telephone line from their institution to a BITNET institution, and agreed to let two or more other institutions connect to them.

The second challenge was to present the information to a person on a terminal in a useful way. Typically, each campus wrote its own email editor, and at Cornell it was Steve Worona who wrote the Cornell editor. It was only sometime later that email editors became standardized.

At the time, there were no vendor-independent, communications protocols for exchanging files between computers. SMTP or Simple Mail Transfer Protocol for an IP-based system wasn't developed until 1982. BITNET was, initially at least, limited to IBM mainframes running RSCS. Even with that restriction, BITNET grew rapidly, and at its peak connected almost 500 organizations throughout the world. Throughout most of the 1980s, it was the lingua franca of the higher-education community.

BITNET, and higher education's role in its growth, was more important than simply being a part of email's evolution. It introduced the utility of email to hundreds of thousands of students, who spread out to other universities and industry, and created marketing demand for what was to become the Internet. With the passage of time, BITNET came to mean "Because Its Time."

The new technology was not without its growing pains. One day some system programmers from the University of Pennsylvania sent Bob Cowles, who was an ace VM system programmer at Cornell, what seemed to be a routine email. The next day when Bob logged onto Cornell's mainframe, everything seemed to be normal until the screen froze and a small PacMan icon begin traveling back and forth across his terminal screen eating the characters displayed. (PacMan was a popular computer game in the early 1980s.) Bob's attempts to enter commands into the system were ignored. After his screen was completely devoured, the bold letters "LOOF," which spelled "fool" backwards, began flashing in his face.

The innocuous email he had received the day before had contained unprintable system commands between the characters of the message. Since those commands were not recognized as characters, they did not display on his terminal screen. They did, however, allow the Penn programmers to gain control of Bob's VM session. They then created a virtual session that initially mirrored a real session, but later switched to the PacMan routine. The joke identified an early crack in our cyber infrastructure.

The Apple Macintosh and the Apple Consortium

In 1983 Stacy Bressler, Apple's regional sales representative, and Dan'l Lewin, Director of Apple's Macintosh Division, brought a prototype Apple Macintosh to Cornell. The machine resembled the computer that would be introduced to the world in January of 1984, but with a lot of parts and wires hanging outside the box. Lewin didn't want to let it out of his sight and carried it with him in a large box. He even went so far as to purchase an extra seat on the airplane for Apple's new entry into the consumer marketplace.

DCS had arranged for Bressler and Lewin to demonstrate the machine to a select group of faculty under a non-disclosure agreement. Dan'l was horrified when he entered the secluded classroom and found almost

two-dozen faculty waiting to see the rumored computer, and stood at the door checking names against the non-disclosure list. Apple took product security very seriously.

The demonstration was successful and the faculty were impressed, particularly with the graphical user interface and mouse. Apple wanted Cornell to commit to being an early adopter of the new computer, in large numbers, in exchange for a significant price break. For the faculty, however, the lack of a programming language more structured than BASIC was a deal breaker. Steve Jobs felt that BASIC was good enough, but finally relented and added Pascal as a programming language when Apple received similar pushback from other universities.

The Macintosh was introduced to the world in January of 1984. It had 128 K of RAM, a 512 x 342 pixel monochrome display, and a single 400 KB 3.5-inch disk drive in a single self-contained plastic cube. The keyboard and mouse were separate. It utilized a Motorola 68000 CPU chip, which was arguably slightly more advanced than the Intel 8086 (actually the slightly variant 8088) CPU used in the IBM-PC. The "Mac" was built in California in one of the industry's first automated factories. They were very proud of their factory and featured it in our visits to corporate headquarters.

The Macintosh "Bundle" made available to students in the spring of 1984 included a Macintosh, as well as writing and drawing software for the then-unheard of price of $1,000. The recommended retail prices were $2,495 for the Mac, $495 for the printer, and $195 for MacWrite/MacPaint.

The Student MacIntosh Bundle
Photograph by Doug Gale

As part of the early adopter arrangement, Cornell became a member of the Apple University Consortium. Originally envisioned as being a half-dozen or so universities, it quickly grew to two dozen at the time of the Macintosh rollout in January of 1984. William Arms' contribution to this incremental book captures some of the excitement at the student rollout.

Apple Consortium, January 1984

Stanford University	Boston College	University of Rochester
Harvard University	Princeton University	University of Pennsylvania
Yale University	Brown University	Northwestern University
Carnegie Mellon University	University of Chicago	University of Notre Dame
University of Michigan	University of Texas	Rice University
Cornell University	University of Washington	City University of New York
Dartmouth College	Reed College	Drexel University
Brigham Young University	University of Utah	Columbia University

The Consortium had regular meetings that were very upbeat, and reflected a mood that we were part of a fundamental change in the way computers were interacting with people. The famous 1984 Superbowl ad captured that mood perfectly. (https://www.youtube.com/watch?v=VtvjbmoDx-I)

I remember two consortium meetings in particular. At one, in Pittsburgh as I recall, I happened to sit at the same dinner table as John Scully and his teenage son, and could not help observing how Scully was grooming his son on how to be a business leader. Coming from a family where I was the first to go to college, I found that quite instructive.

The second was in San Francisco. The reception after the day's events was held in a converted warehouse in which multiple floors, each with multiple rock bands and light shows, opened to a central atrium; the effect was psychedelic. As a long-time fan of classical music, I confess that the atmosphere pulsed with energy.

While the utility of the Mac's graphical user interface seems obvious in retrospect, it was not universally accepted at the time of its introduction. I recall having a lengthy and somewhat acrimonious debate with a colleague at Cornell who felt the interface was a passing fad. We decided to settle the argument by each of us purchasing Apple stock; myself at the going price and he short selling. (I must also disclose that he totally disagrees with my recollection of this event.) Fortunately for him, the price of the stock didn't change much in the months following the introduction. Unfortunately for me, I sold mine months later before it took off.

Institutional Leaders

Who were the institutional leaders during this period? Obviously, the Apple Consortium members were considered leaders by at least one vendor. My personal list, however, would include: Carnegie Mellon University for their work with Project Andrew; Stanford and the University of California-Berkeley for their work in computer networking; Dartmouth for their pioneering work teaching programming and developing BASIC; the University of Michigan as an early leader in creating a state network; CUNY and Yale for their work in developing BITNET; MIT for their work with Project Athena; and of course Cornell.

Microcomputers and Elementary and Secondary Education

Introducing computing into elementary and secondary schools was a slower process, in part because they lacked centralized, information technology units organized to train and support faculty and student end users. In 1982 I was asked, on a consulting basis, to evaluate a draft proposal by the Ithaca Public Schools to their board requesting funding for the acquisition of a large number of Apple II computers. The school system was very unhappy with my report, which faulted the proposal for requesting too much money for hardware and too little for teacher training and support. My report was not included in the presentation that went to their board.

The Downside of Distributed Computing

One unintended consequence of the widespread deployment of microcomputers was to destroy one of the most useful features of timesharing on a mainframe—the ability to exchange information between individual users. We had created islands of isolated computers and users! The only way to move information between microcomputers—assuming they were running the same operating system -- was the physical movement of diskettes, jokingly referred to as "sneakernet."

Sometime in 1980, Fred Hiltz in the Cornell Veterinary School began working on a protocol to allow computers, including microcomputers, to communicate with each other over an asynchronous serial connection. The protocol had the catchy name of "FITS" for FIle Transfer System. The problem was widely recognized nationally, and in February of 1981 EDUNET, an activity of EDUCOM, set up a Task Force to address and consider solutions to the problem. Initially the task force was asked to look at two draft protocols, FITS and a similar protocol being developed at Wisconsin. In the meantime, Alison Brown in DCS had begun working with Fred on actual implementations of the FITS protocol, which by this time was quite well-defined.

I remember traveling to North Carolina to discuss their work with Lou Parker, Jr., the Director of the North Carolina Educational Computing Services, who was also on the task force, and discussing a possible collaboration between Cornell and a similar effort in North Carolina. One morning shortly after returning to Cornell I entered Alison's office and asked her how work was coming on the FITS implementations that she had been working on for months. She answered, "I'm not working on that anymore; I'm doing a KERMIT implementation."

She had learned of a similar file transfer effort at Columbia University, which had been named after Kermit the frog from the Muppet television program; feeling that they were further along in their work than she was, Fred decided to join the Columbia effort. (http://en.wikipedia.org/wiki/Kermit_(protocol)) That egoless spirit of working towards a solution characterized the academic computing community in higher education. For many years Kermit was the de facto standard for computer-to-computer file transfer.

The Emergence of Networking as Strategic

Even with the immediate success of Kermit, it was obvious that the underlying problem of allowing computers and peripherals to talk to each other—irrespective of different hardware and software—remained. The network had become strategic.

At the time, networking at Cornell was largely the responsibility of the Systems and Operations group. Because they were doing an exemplary job, DCS decided to focus on the software needed to get devices on the network. In late 1982 or early 1983 DCS hosted a spaghetti dinner at Alison Brown and Ken Wilson's house. Our pitch to Ken King was that we needed a small VAX running Berkeley Unix both to better support the increasing number of small system users moving to Unix, and to begin exploring software networking protocols, particularly TCP/IP. King's quick agreement makes me suspect that it was the Bison Bridle Principle ("you can lead a bison anywhere you want to as long as it's where he wants to go") and not our spaghetti that led to a quick decision.

Advanced Scientific Computing

The first half of the decade of the '80s also marked a growing national concern that the United States was losing its competitive advantage in high-performance, scientific computing. The 1982 "Report of the Panel on Large Scale Computing in Science and Engineering," also known as the "Lax Report" (http://www.pnl.gov/scales/docs/lax_report1982.pdf), concluded that without new resources "the primacy of U.S. science, engineering, and technology could be threatened relative to that of other countries…" Cornell's Ken Wilson was a member of that panel.

Under the leadership of Wilson and Alex Grimison, Cornell had already begun addressing some of the needs of large-scale computational users by attaching a Floating Point System (FPS) Model 164 array processor to the university's IBM mainframe. Wilson envisioned a field house filled with thousands of array processors. The software challenges associated with mating the two dissimilar architectures and operating systems were formidable, and Wilson became a legend within the Cornell Computing Center's systems group for his ability to overcome the challenges. This effort ultimately led to the Cornell Theory Center, which will be discussed in the next section.

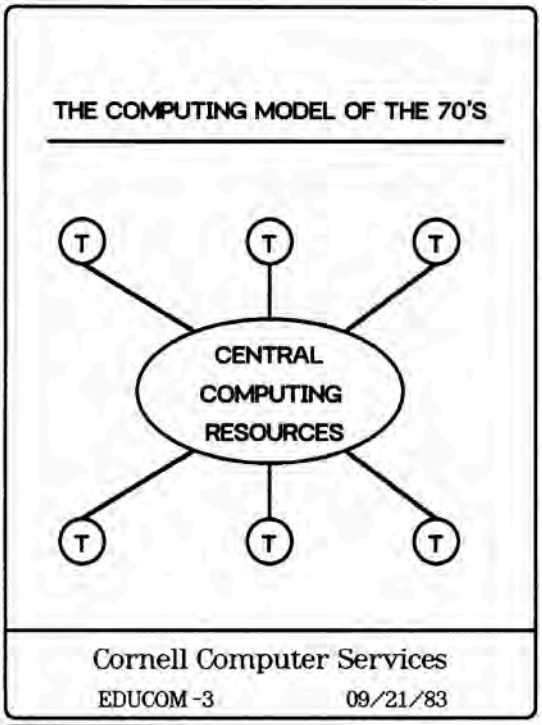
Cornell's Strategy for the Microcomputing Revolution

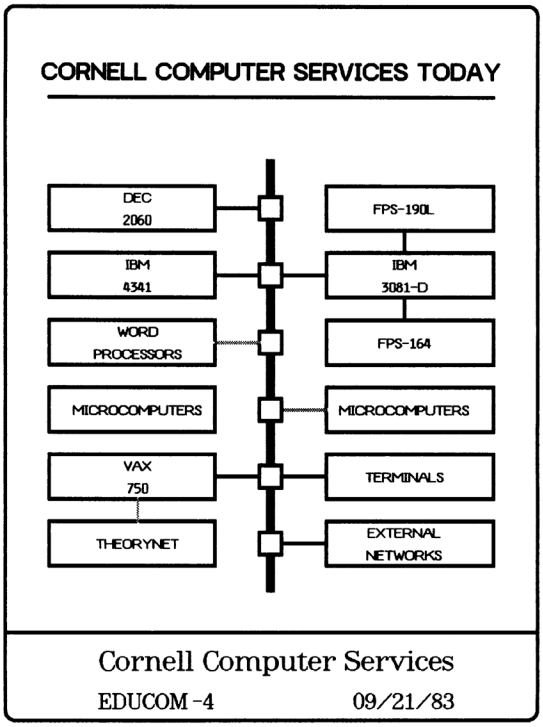
In May of 1983 I was invited to join Edward Friedman, Stevens Institute of Technology, and James Penrod, Pepperdine University, in participating in a plenary session on Personal Computing Implementation at the September EDUCOM meeting. That presentation encapsulates the changes that had occurred at Cornell and a few lead institutions and would shortly occur throughout higher education. (One slightly discouraging aspect

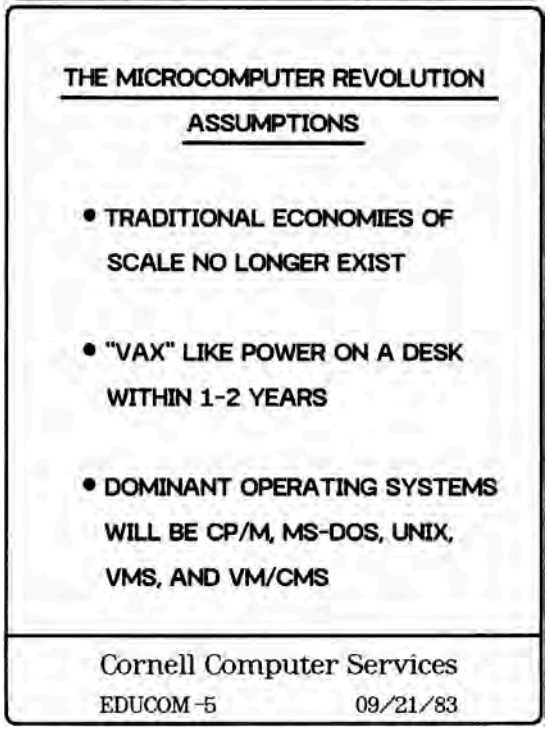
of the presentation was that I got more questions afterwards about how I had prepared my foils (on a Lisa), than about the content of my presentation.) The presentation is illustrative in what we got right and what we got wrong.

Foil "EDUCOM-3" showed the computing model of the 1970s, which was based on sharing centralized, computing resources. Terminals were connected to mainframes with proprietary networks. SNA for IBM, DECnet for Digital Equipment, and XNS for Xerox were typical. There was generally little interoperability between them.

By 1983, as shown on "EDUCOM-4", Cornell had begun making a transition to a single backbone supporting multiple systems. Much of the work developing the interface code for these devices was done at universities in groups such as DCS.



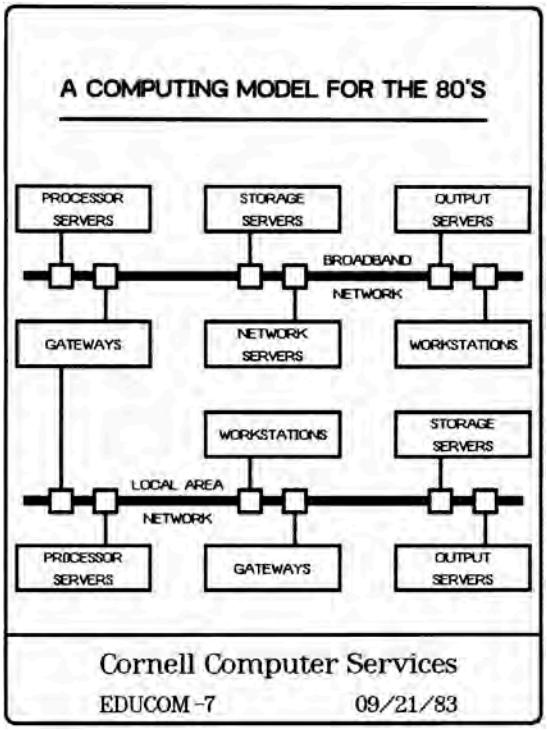


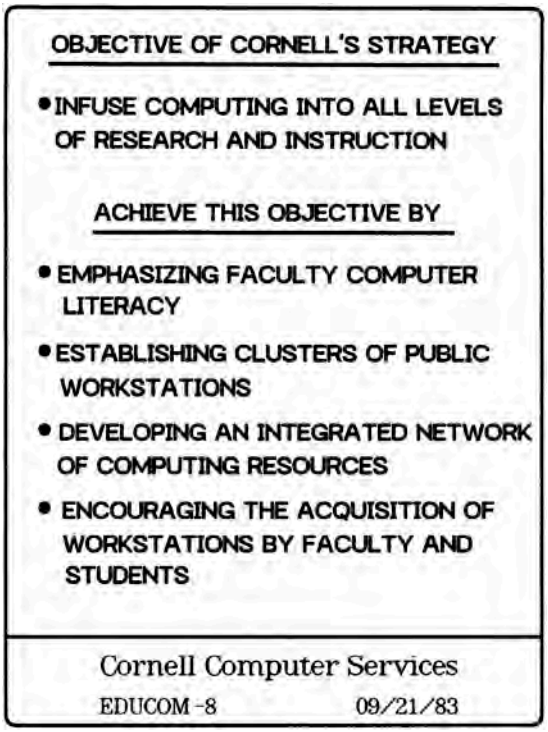


EDUCOM-5, The Microcomputer Revolution Assumptions, was basically an extrapolation of Moore's Law. (The number of transistors per square inch on integrated circuits will double every year.) One consequence of this was that traditional economies of scale no longer existed. Specifically, the cheapest computing was no longer necessarily the sharing of a big machine. Because of Moore's Law and the economies of mass production, the cheapest computing might be multiple smaller machines. Efficiency and keeping a computer running 100% of the time were no longer as important as they once were.

The implications for Cornell, EDUCOM-6, were that microcomputers would be the dominant source of cycles for most users within 2 years and, surprisingly, the total cost of computing to the institution would increase because of a growing user population and more complex systems. Cornell did not regard microcomputers as a silver bullet to decrease institutional computing costs. And, of course, our support mechanisms would have to change radically.

The fifth foil, EDUCOM-7, shows we also clearly understood, earlier than most, that a campus broadband network was going to be the core of a computing model for the 1980s. Our objective and the strategy to achieve those objectives (EDUCOM-8) were fairly common among institutional computing leaders.



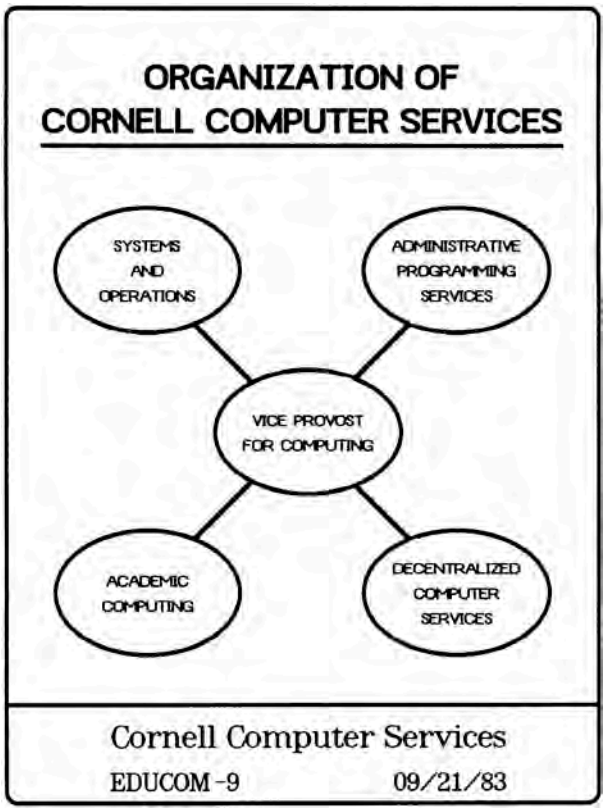


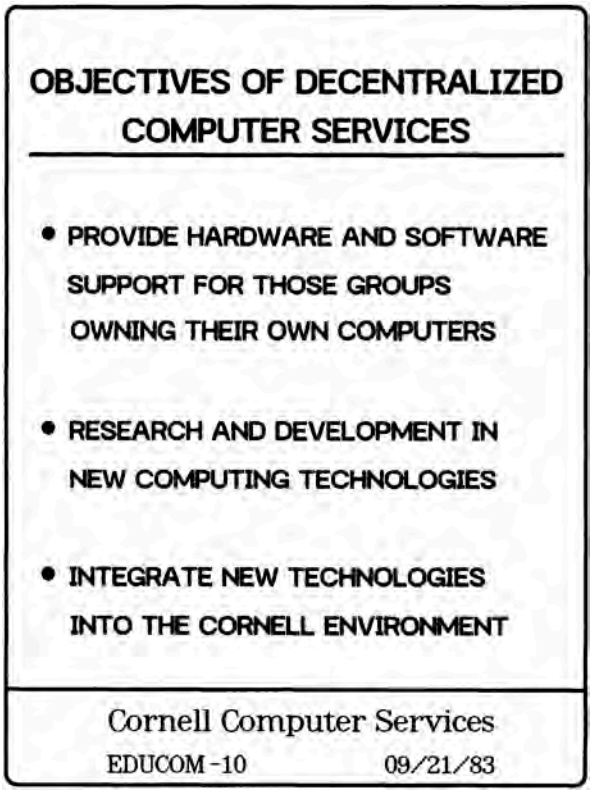
Cornell's Strategy Redux: What We Got Right, What We Got Wrong, and What We Simply Didn't Understand

The technical stuff we got right. Microcomputers did quickly become the dominant source of computer cycles. And a network linking computing resources, from microcomputers to supercomputers, and from printers to scanners, both on campus and nationally, would become the integrating factor.

What we didn't get right was the most efficient organizational structure to support the new realities. Nor did we understand the real role and mission of DCS. We had taken the role of DCS to be "consultation and support for those units owning or planning acquisition of computers." Based on that assumption we created a four-branch organization (EDUCOM-9) consisting of Systems and Operations, Administrative Computing, Academic Computing, and Decentralized Computer Services. What we didn't realize was that the objectives

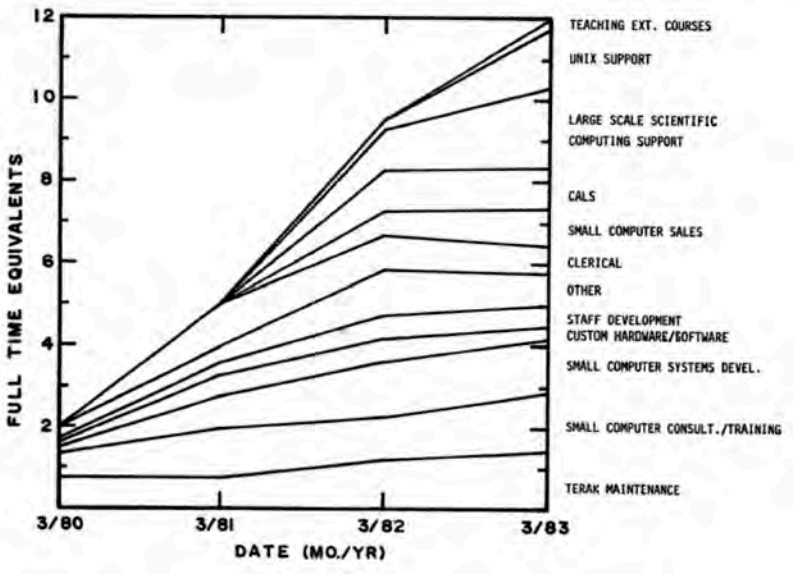
of DCS, shown in EDUCOM-10, were largely crosscutting and applicable to the other three organizational branches as well.





In 1980 I was told by Ken King, I believe, that "DACS has all of the gold and none of the bricks; it should have so much fun that everyone in the organization should want to work there." In short, DACS and later DCS, was a change agent whose purpose was to change the centralized mindset of the organization.

The rotating door within academic computing reflected this interpretation. In May of 1981 Alex Grimison stepped down from the position of Interim Director of Academic Computing to join DCS and to take charge of a scientific computing group. I assumed his role until we were able to recruit a new Director of Academic Computing, Gordon Gallaway. In short, we were having fun in DCS and were functioning more as an internal "skunk works" than an operational unit, such as system and operations. Our mistake was to envision DCS as a permanent organizational structure rather than a temporary change agent.



By the spring of 1983, DCS had grown to 12 FTEs. As shown below, the new functions within DCS were large-scale scientific computing, Unix support (including networking interfaces), and managing small computer sales through external vendors. While the first two would go on to become part of Cornell's "Theory Center" supercomputing initiative, computer sales and small computer support gradually devolved back into more traditional support structures. The centralized culture of the Cornell Computer Center had been changed. A similar organizational

phenomenon occurred in the latter half of the 1990s, when internal "networking skunk works" sprang into existence to drive the networking revolution.

Institutional Collaboration

The early 1980s also marked the growing importance of national organizations facilitating the exchange of information between university and college computing center leaders. It was also marked by a great deal of sharing of experience and knowledge.

In the summer of 1980 I attended a regional meeting, the New England Computing Conference. It was at that meeting that an old hand from either Vermont or New Hampshire passed on to us newbie's the three rules for a computing center director: 1) today is as good a day to die as any other; 2) a horrible end is preferable to horrors without end; and 3) you never control anything. At the time I was amused, but didn't fully appreciate the truth in his words.

The two largest higher-education computing organizations at the time were CAUSE and EDUCOM, but the meetings seldom exceeded several hundred attendees and everyone knew everyone else. A history of both organizations can be found at http://www.educause.edu/about/mission-and-organization/roots-educause. While neither organization was wholly academic nor administrative, in the early years CAUSE seemed to have an administrative computing and "hands-on" feel, while EDUCOM had more of an academic and management/policy feel. In any case I became a regular EDUCOM attendee, although later in the decade I attended both and was on the CAUSE Board when the two organizations merged in 1998. In general there was little overlap in attendees between the two organizations.

The most enjoyable and arguably most useful of the national meetings was the Annual Seminars on Academic Computing held in Snowmass, Colorado. When I first attended in 1982 there were around a hundred attendees. We met in a single small resort hotel, the Mountain Chalet. Attendees frequently brought their families, who would enjoy the Colorado mountains while we attended sessions. Toward the latter part of the decade I would sometimes arrive a few days early and backpack from the Maroon Bells over Buckskin Pass and into Snowmass.

A fourth meeting that I routinely attended was the ACM (Association of Computing Machinery) SIGUCCS (Special Interest Group in University and College Computing Centers) meeting that was held each spring in St. Louis. The meeting was small with a very practical "hands on" feel.

It is hard to describe the excitement and comradery of those early meetings, as compared to the same meetings one or two decades later. In the 1980s I would typically know half or more of the attendees. Computing was just emerging as a discipline, and there was a close bond of togetherness.

Chapter 3: The Late 80s and the Networking Revolution *(not written)*

Chapter 4: The 1990s: The Changing Role of Academic Computing and Networking *(not written)*

Incremental Epilogue-1

Kenneth M. King

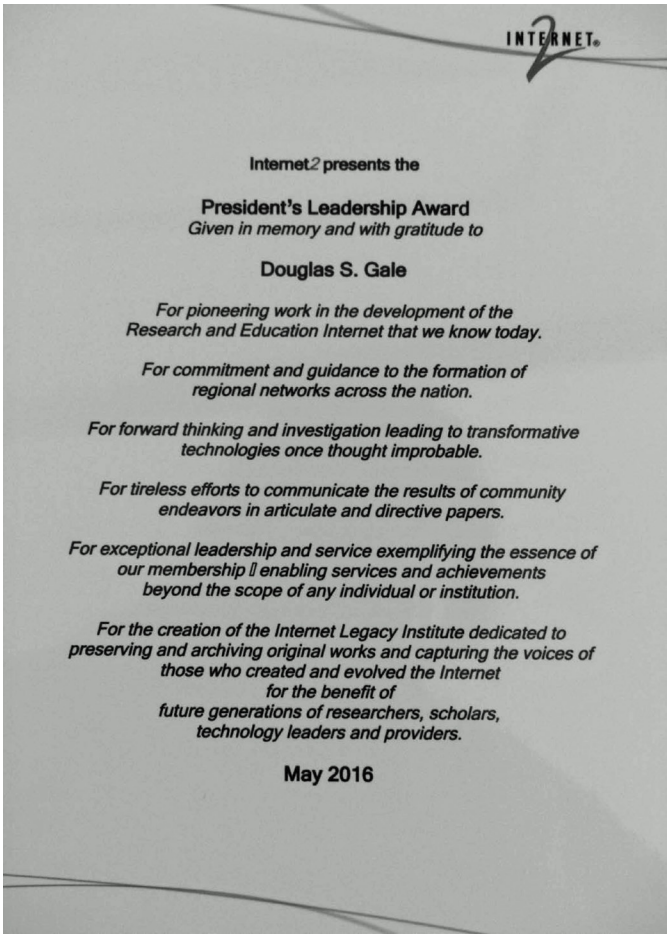
Douglas Shannon Gale died on October 26, 2015, at the age of 73, with his memoir half finished. He intended to write a chapter covering the networking and distributed-computing revolutions that occurred during the late 1980s, and a chapter covering the impact of networking and personal computing on academic computing at universities from the 1990s up to the present. Doug was in the middle of the events that shaped the transformation of computing at universities during and after the 1980s, and he contributed to the evolution of computing and networking in a number of important ways. It is my hope that those who worked with Doug, as I did, will contribute to an effort to extend his memoir by describing his activities and contributions during the years that he would have covered had he been able to complete his memoir. Doug's memoir is part of an incremental book started with the hope that people involved in the early years of academic computing at universities would add their own memoirs, so that that important history will not be lost. Contributions to an effort to complete Doug's memoir will become part of an incremental Epilogue appended to his Memoir.

I met Doug in 1980 at Cornell when he was Director of Decentralized Computing Support. At Cornell in the early '80s, as at most universities, the primary source of computing cycles supporting research and instruction was centrally operated and controlled mainframes. Doug led the effort that resulted in faculty and students largely controlling their own computing devices. This transformation led to an enormous increase in faculty and student productivity and an exponential growth in new computing applications. It also dramatically changed the role of people supporting academic computing at universities. His activities at Cornell are covered in Chapter 2 of his memoir and in a video interview of Doug done a few weeks before he died. This video can be seen at: https://ecommons.cornell.edu/handle/1813/41195

When the formation of NSFNET as a three-tier network was announced in the fall of 1985, Doug was Director of Computing at the University of Nebraska, Lincoln. NSFNET consisted of a backbone network that connected the National Supercomputing Centers; Regional Networks that connected universities to the backbone; and local area networks that connected faculty at universities to a Regional Network. Doug immediately began setting up a Regional Network called MIDnet. MIDnet connected universities in Iowa, Nebraska, Missouri, Oklahoma, and Kansas to NSFNET. MIDnet received an NSF grant in 1986 and shortly later became one of the first Regional Networks to become fully operational. Also in 1986, EDUCOM formed a networking and telecommunications taskforce (NTTF) to advance the connection to NSFNET of every college and university in the country, and later to the world. Doug was very active in this organization and in creating an organization of Regional Networks called FARNET. As NSFNET evolved into the Internet, Doug made major contributions to improving university connectivity at a number of institutions and also served a stint as a program officer at NSF, which was funding network expansion. Overall, Doug was a major player in advancing networking for more than three decades.

Throughout his career Doug meticulously preserved records of meetings and papers devoted to computing and networking. He also had a collection of computers dating back to the earliest days of microcomputers. In 2010, Doug founded the Internet Legacy Institute to preserve and archive information and original source materials about the creation and evolution of the Internet. This organization continues and has made major contributions to the preservation of networking history. Doug travelled extensively with a camcorder to interview people involved with networking during its formative period. His historical record is in the process of being donated to organizations dedicated to preserving historical records. His records will be of enormous value to people seeking to understand events that changed the world.

In May 2016, in recognition of Doug's pioneering efforts in creating the Internet, he posthumously received an INTERNET2 leadership award. At the award ceremony, INTERNET2 President Dave Lambert said: "From the



Internet2 presents the

President's Leadership Award
Given in memory and with gratitude to

Douglas S. Gale

For pioneering work in the development of the Research and Education Internet that we know today.

For commitment and guidance to the formation of regional networks across the nation.

For forward thinking and investigation leading to transformative technologies once thought improbable.

For tireless efforts to communicate the results of community endeavors in articulate and directive papers.

For exceptional leadership and service exemplifying the essence of our membership ▯ enabling services and achievements beyond the scope of any individual or institution.

For the creation of the Internet Legacy Institute dedicated to preserving and archiving original works and capturing the voices of those who created and evolved the Internet for the benefit of future generations of researchers, scholars, technology leaders and providers.

May 2016

beginning, Doug was committed to the development of the network for the R&E community. Whether in recognizing the value of 'dark fiber' when many of his colleagues did not yet appreciate its value, or in his tireless work to bring regional and later national groups together with a common purpose of creating a significant and transformative asset to benefit higher education research and scholarship anywhere in the country, Doug made a difference. Doug was a friend and a mentor to many in our community. When he dedicated himself and his personal resources to the creation of the Internet Legacy Institute it was just an extension of what he had been doing for all of us over the years – making sure the work of the IT R&E Community would stand as a testament of the contributions of thousands of IT professionals who had shaped the global research, scholarship and future of advanced Internet technology."

Below is a picture of the award and a picture of President Dave Lambert presenting the award to Doug's wife and children. From left to right in the family picture are sons Marc and Eric, Henrietta, and Dave Lambert.

Incremental Epilogue-2

Glenn Ricart

I met Doug in the 1980s when he headed computing at the University of Nebraska, Lincoln. I was doing the same at the University of Maryland College Park. I shared with Doug the plans that were evolving for SURAnet to connect the Southeast. I had picked the Southeast as a manageable size because I could fly engineers anywhere and back in the same day. Doug asked who was going to connect the Midwest. I said, "Maybe you will." We agreed to share information, and SURAnet and MIDnet worked closely to mirror policies and methods to get more than one-third of the states connected to what became known as the Internet. Gale's MIDnet became the first fully operational regional network.

Doug continued to be a connector, paying the favor forward and helping many other regional networks get their own starts. He was also instrumental in helping Richard Mandelbaum and me found FARNET, the Federation of American Research Networks. Doug was always willing to share information, and he became a great enabler of many other state and regional networks around the country.

Doug continued to be an important promoter and connector as Program Director for the National Science Foundation's program to support the National Research and Education Network (which is what we called the Internet at that time). As Program Director, he oversaw a vast expansion of the network for science and education, but without limiting the network to those particular uses.

In his later years, Doug became a chronicler of the early Internet, founding the Internet Legacy Institute. The Internet Legacy Institute has perhaps the largest collection of early Internet papers, artifacts, and electronic records. This alone is a major achievement. Because Doug was always so friendly and helpful, other players were happy to share their materials with him.

Incremental Epilogue-3

Ann O'Beay

Doug Gale – from Cornell, to MIDnet, to NSF, to OARnet, to the Internet Legacy Institute (ILI) – was instrumental in the design and development of the Internet, with exceptional achievements that impacted the Internet's advancement and evolution. Throughout his exemplary career he not only personally left an indelible mark on Internet history, but also assured its preservation.

Doug was a person of action. When he saw a need, he envisioned and created innovative solutions and championed others to optimize outcomes. I first met Doug during the NSFNET era while I was at MCI – and MIDnet was one of the very first regional networks to connect to the NSFNET. When my path led me to the CTO position at the Ohio Board of Regents in 2010, people in Ohio were still talking about his pioneering and inspirational role in advancing the work of Ohio's R&E network (OARnet.)

Through ILI, Doug tirelessly gathered and preserved over 100 personal accounts of Internet Pioneers through video and oral history recordings (including those of a number of esteemed Internet Hall of Fame inductees), as well as archiving an unparalleled collection of early Internet papers and artifacts, which he had meticulously collected and chronicled over the decades.

I had the opportunity to sit in when Doug was conducting a number of audio and video recordings for ILI. He naturally knew how to draw out stories from those interviewed in the context of the remarkable journey

Internet pioneers collectively share. He gave tirelessly of his time and committed significant personal resources to the quest. The materials and recordings Doug collected are already being referenced for books and scholarly publications in ways that help us learn from the past to inform the future. ILI has among its treasures resources that, without Doug's vision, initiative, and persistence, would have otherwise been lost to the world.

Incremental Epilogue-4

Jim Williams

Doug was an early and instrumental contributor to the evolution of the US NSFNET and its transition to the Internet we enjoy today. Moreover, Doug devoted his last decade to chronicling and preserving the contributions of others as the founder, leader, and primary collector for the Internet Legacy Institute (ILI).

I have known and had the pleasure of working with Doug from the mid-1980s until his death in 2015. In the beginning, we were both actively leading efforts to connect regional institutions of higher education to the emerging NSFNET, in order to provide access and communication to and among resources primarily for the scientific community. Doug's pioneering efforts as creator, principal investigator, and architect of MIDnet – one of the first three regional networks to link institutions to the NSFNET in 1987 – preceded my related efforts in Nevada, and Doug was a valuable source of counsel and inspiration to me. This was a time when getting resources for these initiatives was not easy, as the world and our own constituents had little or no grasp of what it was we were trying to do, and how it might benefit them – let alone the globe. The established communications providers thought of "packet switching" as an academic novelty that had little probability of success.

Our paths continued to cross throughout the years – particularly when Doug provided leadership and vision at the NSF as Director of the NSFNET Program, while I was serving as Director of National Networking at Merit Network. We worked together again when he was engaged as an active and influential member of the Federation of American Research Networks (FARNET). Our last collaboration began when he coerced me to contribute to the workings of the ILI, an effort that continues.

Throughout Doug's career, he maintained a steady focus on influencing the development of what we now know as the Internet, in a way that would provide positive benefits to society. He foresaw the possibility of many of the applications we use today. He was an innovator in his own right, and an inspiration to others. His constant positive attitude was contagious. In my view, his efforts throughout his career made substantial contributions to creating the Internet and to many fellow pioneers who participated in that effort.